CPC464

Benutzerhandbuch

AMSTRAD

CPC464 COLOUR PERSONAL COMPUTER 64K

Vorbemerkung zu diesem Benutzerhandbuch

Niemals zuvor in der Geschichte der technischen Erfingungen kamen in einem Bereich in so kurzer Zeit so viele Fortschritte und Neuentwicklungen heraus, wie in der elektronischen Datenverarbeitung.

Es liegt im Wesen dieser faszinierenden Technologie begründet, daß **eine** Neuerung oder Erkenntnis bereits die weiteren vorbereitet und ermöglicht. Ein rasanter Fortschrittsprozeß also, der auf einmalige Weise die Kreativität der Menschen weckt und ihre Ideen und Wünsche verwirklicht.

So kann Ihnen auch das hier vorliegende Benutzerhandbuch für den CPC464 nicht alle seine Hardwareeigenschaften, die Vollständigkeit seines Betriebssystems und den kompletten Umfang seines BASIC aufzeigen – dies würde Bände füllen.

Mit dieser Anleitung wollen wir Sie in die Arbeit mit dem Amstrad CPC464 und seine Software einführen. Es liegen gesondert weitere Lehrgänge und Informationen vor.

Wenn Sie schon andere BASIC-Dialekte kennen, werden Sie „Amstrad BASIC" schnell beherrschen. Doch auch für Einsteiger ist es gut zu wissen, daß Ihnen eine klar verständliche und eindeutige Terminologie den Umgang mit dem CPC464 erleichert.

Dazu gehören auch Real-Time-Funktionen, die bisher bei so preiswerten Computern nicht verfügbar waren.

Dieses Benutzerhandbuch ist in 3 Teile gegliedert:

Der erste Teil führt den Neuling in die Welt der Datenverarbeitung ein und erklärt die Konzepte, Begriffe und Möglichkeiten.

Wer bisher noch keinen Personalcomputer besaß oder noch keine Gelegenheit hatte, damit umzugehen, sollte diesen ersten Teil durcharbeiten.

Teil 2 beinhaltet die eigentliche Bedienungsanleitung für den CPC464 und seine spezifischen Vorteile.

Wer schon Erfahrung hat und die Begriffe der Datenverarbeitung kennt, kann gleich mit diesem Teil, Kapitel 1, beginnen.

Vervielfältigung und Weitergabe von Informationen aus diesem Benutzerhandbuch – auch auszugsweise – bedürfen der vorherigen schriftlichen Zustimmung durch Amstrad .

Alle technischen Daten, Informationen sowie Eigenschaften des in diesem Benutzerhandbuch beschriebenen Produktes wurden nach bestem Wissen zusammengestellt und entsprechen dem Stand bei Drucklegung.

Änderungen und Verbesserungen des Produktes aufgrund technischer Neuentwicklungen sind möglich.

Eine Zusammenstellung aller Veränderungen und etwaiger Korrekturen dieses Benutzerhandbuchs sind bei Amstrad erhältlich.

Alle Wartungs- und Servicearbeiten müssen von Amstrad—autorisierten Händlern durchgeführt werden. Amstrad trägt keine Verantwortung für Schäden, die durch unsachgemäße Wartung bzw. Service durch unbefugte Personen entstanden sind. Diese Anleitung dient nur dazu, dem Anwender bei der Benutzung des Produktes zu helfen. Amstrad übernimmt keine Verantwortung für Schäden, die durch die Anwendung von falschen Informationen, oder Fehlern bzw. fehlenden Informationen in dieser Anleitung, oder durch eine falsche Anwendung des Produktes verursacht wurden.

Alle Hinweise auf Z80 in dieser Anleitung wurden im Einvernehmen mit Zilog Inc. gemacht.

Erste Ausgabe 1984

Originalausgabe in Englisch

Original Copyright © 1984 AMSOFT, AMSTRAD Consumer Electronics plc and Locomotive Software Limited

Deutsche Übersetzung: Peter Eschenbacher, Wulf-Dietrich Wenzel

Wiederholungen mancher Grundsätze dienen dazu, die notwendigen Handgriffe im Umgang mit BASIC besser zu erlernen und den unmittelbaren Einstieg in die Arbeit mit Ton und Graphik zu erleichtern.

Teil 3 schließlich ist ein ausführlicher Anhang mit einer Übersicht über die Konzepte der Datenverarbeitung sowie gerätespezifische Erläuterungen.

Wir empfehlen Ihnen, begleitend zu diesem Benutzerhandbuch die programmierte Unterweisung „Selbstlern-BASIC" von Amstrad zu erwerben und durchzuarbeiten.

Sie bietet Ihnen eine intensive und umfassende Einführung in die Arbeit mit dem CPC464 und vermittelt Ihnen die nahezu grenzenlosen Möglichkeiten des Systems als normaler Computer, als Lehrer oder Spielpartner.

Auch in der Welt der Datenverarbeitung gilt das Sprichwort „Probieren geht über studieren". Sie lernen die Datenverarbeitung am einfachsten im Umgang mit einem Computer. Und Sie erkennen die Chancen, die er Ihnen persönlich und beruflich eröffnet am besten, indem Sie ihn fordern, oder sich von ihm fordern lassen.

Bald werden Sie feststellen, daß Ihnen – gemessen an seinem Preis – kein anderer Computer mehr Möglichkeiten bietet als der CPC464 von Amstrad . Zudem ist er sehr bequem zu bedienen. Übrigens: Wenn Sie Ideen, Vorschläge und Verbesserungen zum Produkt (Hardware oder Software) haben, schreiben Sie uns. Über jede Anregung würden wir uns freuen.

Wir wünschen Ihnen nun viel Freude und Erfolg.

WICHTIG!

Wenn Sie dieses Benutzerhandbuch lesen, sollten Sie auf die verschiedenen Druckarten achten. Sie zeigen, in welcher Weise ein Bezug auf Programme gemacht wird. Z. B. [TASTEN] am Computer, die kein Zeichen am Bildschirm verursachen. Oder allgemeine Kommentare, die als Hinweise dienen, jedoch nicht als Befehl eingegeben werden.

1. Netzstecker immer an einer geerdeten Steckdose anschließen, wie unter „Aufstellen" beschrieben.
2. Niemals Tastatur, Monitor oder Stromversorgung/Modulator an irgendeine andere Stromversorgung oder andere Geräte anschließen, die nicht in diesem Handbuch beschrieben sind. Die Geräte können schwer beschädigt werden und die Garantie erlischt.
3. Blumenvasen, Getränke, usw. weit entfernt von Tastatur, Monitor und Stromversorgung/Modulator halten. Durch Flüssigkeit werden in diesen Geräten große Schäden verursacht. In einem solchen Fall müssen qualifizierte Wartungstechniker gerufen werden.
4. Lüftungsöffnungen oben und hinten an Tastatur, Monitor und Stromversorgung/Modulator dürfen nicht zugedeckt werden.
5. Wenn Sie den Strom abschalten, gehen alle Informationen im Speicher des CPC464 verloren. Im Kapitel 2 wird beschrieben, wie ein Programm gesichert werden kann. Vorher die Grundsatzbeschreibung lesen.
6. Es wird empfohlen, nur Computer-taugliche Cassetten zu benutzen. Es ist jedoch möglich, Audio-Cassetten guter Qualität zu benutzen (Ausnahme CrO_2, ‚Metal' und C-120-Spieldauer).
   Um ein Programm schneller zu finden, empfehlen wir, C-12 Cassetten (6 Minuten je Seite) zu benutzen.
7. Cassetten mit Programmen anderer Computer können am CPC464 weder geladen werden noch ablaufen.
8. Die Schreibtaste (RECORD) kann nicht gedrückt werden, wenn die Sicherungsstreifen an der Cassette fehlen. Dies verhindert ein ungewolltes Überschreiben. Bitte diese Taste nicht mit Gewalt drücken, es können Schäden am Gerät entstehen. Um eine so gesicherte Cassette wieder zu beschreiben, müssen die Öffnungen mit Klebestreifen abgedeckt werden.
9. Sorgen Sie dafür, daß die Cassette den Vorspannstreifen passiert hat, bevor ein Programm gesichert wird.
10. Setzen Sie die Geräte nicht praller Sonne aus, auch nicht extremer Hitze, Kälte, Feuchtigkeit, Staub oder Erschütterungen. Cassetten sollen niemals in der Nähe von starken magnetischen Quellen stehen, z. B. bei Lautsprechern oder großen elektrischen Motoren.
11. Gute Behandlung Ihrer Cassetten und regelmäßige Säuberung Ihres Datacorders sorgen für eine fehlerfreie Sicherung und schnelles Wiederauffinden der Programme.
12. Geräte nicht öffnen. Sie enthalten keine vom Anwender reparierbaren Teile. Wartung von qualifizierten Technikern durchführen lassen.
13. Weder alle noch ein Teil der Information, die hier enthalten sind, dürfen in irgendeiner Weise modifiziert oder kopiert werden. Auch nicht die hier beschriebenen Programme oder Produkte.

INHALTSVERZEICHNIS

Zu diesem Benutzerhandbuch

Grundlagenkurs für Anfänger

Eine behutsame Einführung für den Datenverarbeitungs-Neuling
G1 Aufbau
G2 Tastatur
G3 Graphik, Darstellungsarten und Musik

1 Start

Anschließen des Computers
Einschalten
Tastatur
Anzeigen des Zeichenvorrates
Editieren am Bildschirm

2 Datacorder

Laden und Sichern mit dem Datacorder
Die 'Welcome'-Cassette

3 BASIC Einführung

Einführung in die Grundzüge von CPC464 BASIC
Syntax des Amstrad BASIC
Variablen, Operatoren
Einfache BASIC Übungen
Frei definierbare Tasten
PRINT und formatierte Anzeige

4 Variablen, Operatoren und Daten

Formatierte Anzeige
Daten und Matrizen
Dimensionieren
Positionieren

5 Graphik Einführung

Das Wesen der CPC464 Farb-Graphik
INK, PEN, PAPER
MODES, PIXELS, ORIGINS, WINDOWS
Einfache Graphik-Routinen
Selbst-definierbare Zeichen

6 Musik Einführung

Ton-Umfang des CPC464
Hüllkurven für Klang und Lautstärke
Ton-Warteschlangen
Effekte

7 Drucker und Joysticks

Wie man Joysticks benutzt
Der JOY-Befehl
Anschluß eines Parallel-Druckers

8 Nachschlageteil für Amstrad BASIC

Eine kurze Übersicht über die Befehle BASIC und die Schlüsselwörter

9 Weitere Information zur Programmierung

Die interne Organisation der Programme – Firmware
Unterbrechungen und ihre Bedeutung
Steuerzeichen
Die Beziehung zwischen Unterroutinen in Maschinencode und den höheren BASIC-Befehlen

10 Unterbrechungsmöglichkeiten

Die Echtzeit Funktionen
AFTER,EVERY und REMAIN

ANHANG

I Für den Anfänger: Was man von einem Computer erwarten kann und und was nicht
Wörterbuch der Datenverarbeitungsbegriffe

II Bits und Bytes, Binäres und Hexadezimales

III ASCII Zeichenvorrat
Zeichendefinitionen und Matrix
Tastaturzeichen und Erweiterungen

IV Einführung und Überblick für Fortgeschrittene

V User-Interface und Expansion-Bus
Ein-/Ausgabe Anschlüsse

VI Bildschirm-Formate planen und organisieren

VII Planung der Musik
Noten und Klangperioden

VIII Reservierte Wörter
Fehlercodes und Meldungen

Amstrad CPC464 Einführungskurs für Anfänger Grundlagen 1: AUFSTELLEN

Hinweise zum Auspacken, Zusammenstecken und Einschalten Ihres CPC464 Systems.

Der Amstrad CPC464 Colour Personal Computer kann wahlweise betrieben werden mit:

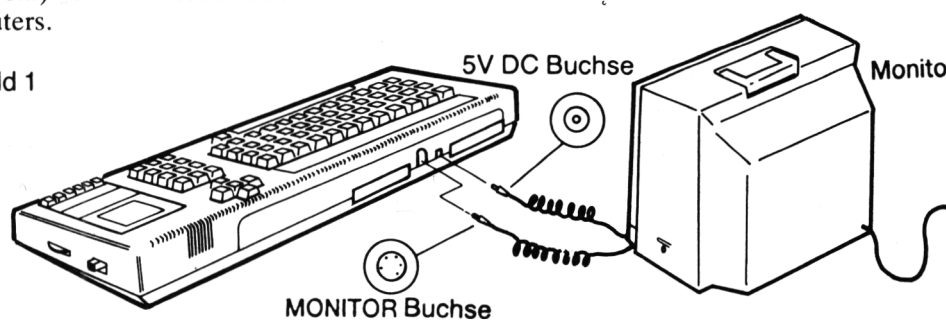
1.1 Amstrad GT65	**Grünmonitor**
1.2 Amstrad CTM644	**Farbmonitor**
1.3 Amstrad MP2	**Modulator/Stromversorgung und einem normalen Fernsehgerät.**

Schauen Sie sich jetzt bitte den entsprechenden Abschnitt an, damit Sie Ihr Computersystem richtig anschließen.

1.1 Amstrad GT65 Grünmonitor

Packen Sie den Monitor aus. Stellen Sie den Monitor auf einen Tisch in der Nähe einer Steckdose. Der Tisch sollte so groß sein, daß auf ihm der Computer bequem vor dem Monitor Platz hat. Stecken Sie jetzt das Kabel mit dem größeren Stecker (6pin DIN), das vom Monitor kommt, in die mit `MONITOR` bezeichnete Buchse an der Computerrückseite (siehe Bild 1). Verbinden Sie das Kabel mit dem kleineren Stecker (Gleichstrom) vom Monitor mit der `5V DC` bezeichneten Buchse auf der Rückseite des Computers.

Bild 1



Sorgen Sie dafür, daß die Einschalttaste (POWER) am Monitor ausgeschaltet (OFF) ist. Stecken Sie den Netzstecker in eine 220 Volt (Wechselstrom) Steckdose.

Schalten Sie jetzt den Monitor ein, und schalten Sie dann den Computer mit dem Schiebeschalter ein, der auf der rechten Seite des Computers (POWER) ist.

Die rote Lampe (ON) auf der Computer-Tastatur, oben Mitte, leuchtet jetzt auf und am Monitor erscheint folgendes Bild:

```
Amstrad 64K Microcomputer (v1)

© 1984 Amstrad Consumer Electronics plc
          and Locomotive Software Ltd.

BASIC 1.0

Ready
■
```

Cursor

Um die Augen nicht unnötig zu belasten, Helligkeitsregelung (BRIGHTNESS) so einstellen, daß weder eine Spiegelung noch eine Verzerrung der Schrift vorkommt.

Die Kontrasteinstellung (CONTRAST) möglichst niedrig einstellen.

Die Vertikalsteuerung (VHOLD) am GT65 so einstellen, daß das Bild richtig in der Mitte des Bildschirms steht und weder durchläuft noch zittert.

ACHTUNG!

Wenn das System nicht benutzt wird: Stecker ziehen.

Da intern nichts verbunden werden muß, sollten Sie nicht versuchen, die Geräte zu öffnen.

1.2 Amstrad CTM644 Farbmonitor

Packen Sie den Monitor aus und stellen Sie ihn auf einen Tisch in der Nähe einer Steckdose. Stellen Sie den Computer vor den Monitor auf den Tisch.

Verbinden Sie, wie im Bild 1 auf der vorherigen Seite gezeigt, das Kabel mit dem größeren Stecker (6pin DIN), das vom Monitor kommt, mit der mit `MONITOR` bezeichneten Buchse auf der Computer-Rückseite. Stecken Sie das Kabel mit dem kleineren Stecker (Gleichstrom), das vom Monitor kommt, in die mit `5V DC` bezeichete Buchse auf der Computer-Rückseite.

Vergewissern Sie sich, daß die Einschalttaste am Monitor auf Aus `(OFF)` steht. Stecken Sie den Netzstecker des Monitors in eine 220 V (Wechselstrom) Steckdose.

Schalten Sie jetzt den Monitor ein, und schalten Sie dann den Computer mit dem Schiebeschalter ein, den Sie auf der rechten Seite des Computers finden `(POWER)`.

ACHTUNG!

Wenn das System nicht benutzt wird: Stecker ziehen.

Da intern nichts verbunden werden muß, sollten Sie nicht versuchen, die Geräte zu öffnen.

Die rote Lampe (ON) auf der Computer-Tastatur, oben Mitte, leuchtet jetzt auf und am Monitor erscheint folgendes Bild:

```
Amstrad 64K Microcomputer (v1)

© 1984 Amstrad Consumer Electronics plc
          and Locomotive Software Ltd.

BASIC 1.0

Ready
■
```

Cursor

Um die Augen nicht unnötig zu belasten, Helligkeitsregelung (BRIGHTNESS) so einstellen, daß weder eine Spiegelung noch eine Verzerrung der Schrift vorkommt.

1.3 Amstrad MP2 Modulator/Stromversorgung und normales Farbfernsehgerät (UHF)

Das MP2 ist ein Zusatzgerät, das Sie vielleicht dazu gekauft haben, wenn Sie einen CPC464 mit einem grünen GT65 -Monitor haben. Mit Hilfe des MP2 können Sie Ihren Computer mit Ihrem Farbfernsehgerät zu Hause verbinden, und können so die ganzen Farbeigenschaften des CPC464 ausnutzen.

Packen Sie das MP2 (Modulator/Stromversorgung) aus und stellen Sie es rechts vom Computer auf einen geeigneten Tisch in der Nähe des Fernsehgerätes und einer Steckdose.

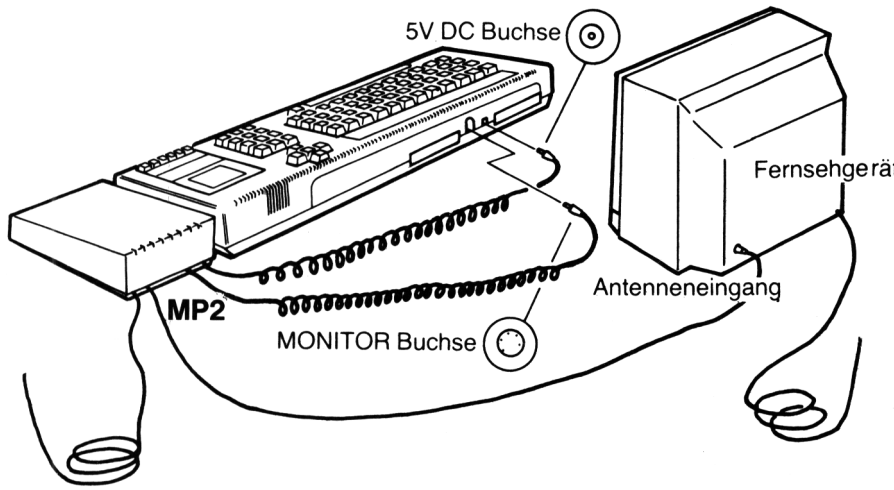


Bild 2: Das MP2 – Modulator und Stromversorgung

Verbinden Sie, wie in Bild 2 gezeigt, das Kabel mit dem größeren Stecker (6pin DIN), das vom MP2 kommt, mit der MONITOR Buchse auf der Rückseite des Computers. Stecken sie den kleineren Stecker (Gleichstrom) des MP2 in die mit 5V DC bezeichnete Buchse auf der Rückseite des Computers.

Verbinden Sie das Kabel mit dem Antennenstecker des MP2 mit dem Antenneneingang Ihres Fernsehgerätes.

Vergewissern Sie sich, daß der Netzschalter (POWER) auf der rechten Seite des Computers auf OFF steht und stecken Sie dann den Netzstecker des MP2 in die Steckdose.

Drehen Sie die Lautstärke am Fernseher ganz zurück, schalten Sie den Fernseher ein und dann erst den Computer mit dem Schiebeschalter (POWER) auf der rechten Seite.

Die rote Lampe ON auf der Computer Tastatur, oben Mitte, sollte jetzt aufleuchten, und Sie müssen jetzt Ihren Fernseher so einstellen, daß ein Signal vom Computer empfangen wird.

Falls Sie einen Fernseher mit Programmtasten besitzen, drücken Sie eine freie bzw. unbenutzte Taste. Stellen Sie den Fernseher lt. Bedienungsanleitung so ein, daß ungefähr Kanal 36 empfangen wird, und Sie folgendes Bild bekommen:

```
Amstrad 64K Microcomputer (v1)

© 1984 Amstrad Consumer Electronics plc
          and Locomotive Software Ltd.

BASIC 1.0

Ready
■
```

Cursor

Justieren Sie den Fernseher bis das Bild klar erscheint. Die Schrift sollte dann gold/gelb auf einem tiefblauen Hintergrund sein.

Falls Ihr Fernseher einen Programmdrehwähler hat, drehen Sie den Schalter bis das Bild (oben) erscheint und absolut ruhig bleibt (wiederum ungefähr bei Kanal 36).

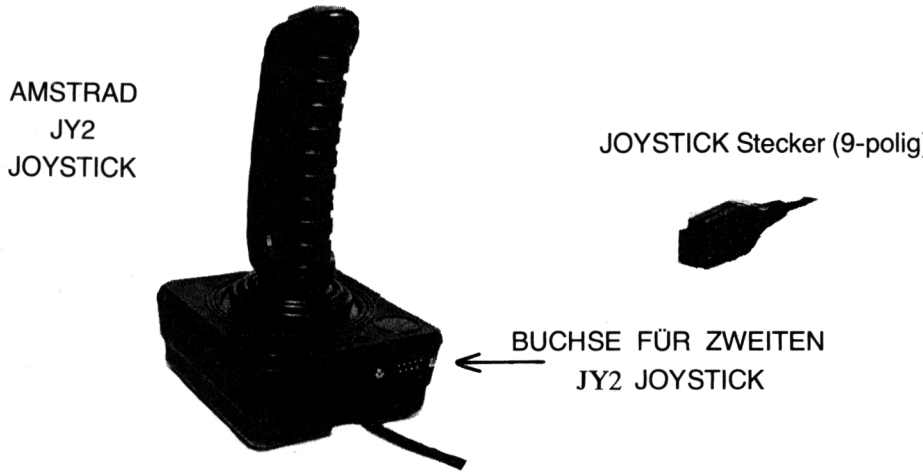
ACHTUNG!

Ziehen Sie den Netzstecker, wenn Sie nicht mit dem System arbeiten.

Da Sie intern nichts anschließen müssen, sollten Sie die Geräte auf keinen Fall öffnen.

1.4 Joystick

Den Amstrad Joystick JY2 können Sie als Zubehör dazu kaufen, wenn Sie auf Ihrem CPC464 Computerspiele einsetzen, die mit einem Joystick gespielt werden und bei denen man auch „schießen" kann. Der JY2 wird an der 9-poligen Buchse an der Linken Seite des Computers angeschlossen, die mit **USER PORTS (I/O)** bezeichnet ist. Der Amstrad CPC464 Computer kann mit zwei Joysticks benutzt werden. Der zweite Joystick JY2 kann in eine Buchse im ersten Joystick eingesteckt werden.



1.5 Welcome Cassette

In der einen Seite der Styropor-Verpackung Ihres Computers haben Sie die „Welcome" Cassette gefunden. Öffnen Sie den Deckel des Datacorders, indem Sie die Taste `[STOP/EJECT]` drücken, und legen Sie dann die Cassette in den Datacorder Ihres Computers, mit der Seite 1 nach oben, wie in Bild 3 gezeigt:

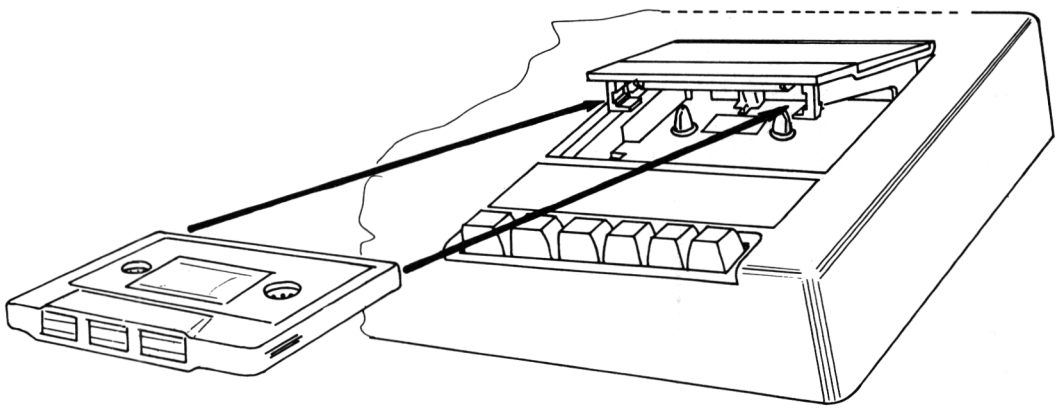


Bild 3: So legen Sie eine Cassette richtig in den Datacorder ein

Schließen Sie den Deckel, bis er einrastet, und drücken Sie dann die `[REW]` Taste des Datacorders, damit die Cassette auch sicher an den Bandanfang zurückgespult ist. Sobald das Band stoppt, drücken Sie die `[STOP/EJECT]` Taste. Stellen Sie das Bandzählwerk auf 000, indem Sie auf den `COUNTER RESET` Knopf drücken.

Drücken Sie auf die [CTRL] Taste (Kontrolle), und *GLEICHZEITIG* auf die kleine [ENTER] Taste rechts unten im Ziffernblock neben dem Datacorder. Am Bildschirm erscheinen die Anweisungen:

```
RUN"
Press PLAY then any key:
```

Drücken Sie jetzt auf die [PLAY] (Spielen) Taste vorn am Datacorder bis sie einrastet, dann drücken Sie eine beliebige Buchstaben-, Nummern-, oder die Leertaste, oder die [ENTER] Taste.

Die Cassette dreht sich, und nach kurzer Zeit erscheint am Bildschirm die Meldung:

```
Loading WELCOME 1 block 1
```

Das Band hat ca. 5 Minuten Ladezeit. Sie können am Bildschirm verfolgen wie sich die Blocknummer in 2,3, usw. verändert, bis die Cassette anhält. Jetzt fängt das ‚Welcome' Programm an zu laufen. Setzen Sie sich hin und schauen Sie zu. Das Programm läuft immer wieder von vorn. Wenn Sie aufhören möchten, drücken Sie zweimal auf die [ESC]Taste. Das Programm hält an und Sie können die [STOP]Taste drücken, die Cassette entnehmen, umdrehen und Seite 2 einlegen.

Nachdem Sie die Cassette auf Seite 2 eingelegt haben, denken Sie daran, die [REW] Taste zu drücken, so daß Sie sicher sein können, daß die Cassette am Anfang steht.

Drücken Sie jetzt die [CTRL] Taste und *GLEICHZEITIG* auf die kleine [ENTER] Taste rechts unten im Ziffernblock neben dem Datacorder. Am Bildschirm erscheinen die Anweisungen:

```
RUN"
Press PLAY then any key:
```

Drücken Sie auf die [PLAY]Taste am Datacorder bis sie einrastet, dann auf eine beliebige Buchstaben-, Nummern-, oder die Leertaste, oder die [ENTER] Taste.

Die Cassette dreht sich und nach kurzer Zeit erscheint die Meldung am Bildschirm:

```
Loading WELCOME 2 block 1
```

Folgen Sie den Anweisungen am Bildschirm und das Programm lädt Sie ein mitzumachen, indem Sie Befehle eingeben, während das Programm abläuft.

1.6 Laden anderer Software-Cassetten

Das „Welcome“ Band kann nur so geladen werden und ablaufen, wie es im vorherigen Abschnitt (1.5) beschrieben wurde. Ungeschützte (unprotected) BASIC-Programme können jedoch auch anders, wie folgt geladen werden. Spulen Sie das Band, das Sie eingelegt haben, zurück (`[REW]` Taste am Datacorder), bis sich die Spulen nicht mehr drehen, und drücken Sie dann gleich die `[STOP/EJECT]` Taste.

Setzen Sie den Computer zurück, um seinen Speicher zu löschen, indem Sie nacheinander die `[CTRL], [SHIFT]` und `[ESC]` Taste drücken, die jeweiligen Tasten dabei aber gedrückt halten, bis Sie die `[ESC]` Taste gedrückt haben. Der Bildschirm wird gelöscht und die Startmeldung erscheint, so als ob Sie gerade eingeschaltet hätten. Der Ausdruck `[ENTER]` in den folgenden Anweisungen bedeutet, daß Sie auf eine der beiden `[ENTER]` Tasten drücken sollen, und nicht das Wort `[ENTER]` eintippen! Das " Zeichen erhalten Sie, wenn Sie gleichzeitig auf eine der `[SHIFT]` Tasten und auf die Taste 2 in der obersten Tastenreihe drücken.

Tippen Sie ein:

```
load "" [ENTER]
```

Der Computer sagt Ihnen ...

```
Press PLAY then any key:
```

Drücken Sie jetzt auf die `[PLAY]` Taste am Datacorder bis sie einrastet, dann auf eine beliebige Buchstaben-, Nummern-, oder die Leertaste, oder die `[ENTER]` Taste.

Das Band dreht sich, und nach kurzer Zeit erscheint folgende Meldung am Bildschirm:

```
Loading <programmname> block 1
```

Die Blocknummern erhöhen sich bis die Cassette geladen ist und die Meldung:

```
Ready
```

... am Bildschirm erscheint.

Sie können auch den Programmnamen explizit angeben, indem Sie eingeben:

```
load "<titel>" [ENTER]
```

<titel> ist der Programmname

Der Computer sagt Ihnen ...

```
Press PLAY then any key:
```

Drücken Sie jetzt auf die [PLAY] Taste am Datacorder bis sie einrastet, und dann auf eine beliebige Buchstaben-, Nummern-, oder die Leertaste, oder die [ENTER] Taste.

Das Band dreht sich. Falls das gewünschte Programm nicht am Anfang der Cassette steht, sucht der Computer die ganze Cassette durch, bis er den gleichen Titel findet, den Sie laden möchten. Achten Sie darauf, daß der Programmtitel richtig eingegeben wird.

Falls der Computer andere Titel findet, während er nach dem gewünschten sucht, erscheint die Meldung am Bildschirm:

Found <*anderer titel*> block 1

Der Computer lädt dieses Programm nicht, sondern sucht weiter auf dem Band bis er den richtigen Titel findet, oder Sie auf die [ESC] Taste drücken, um das Suchen zu beenden.

Wenn das Programm gefunden wurde, erscheint die Meldung:

Loading <*titel*> block 1

Die Blocknummern werden hochgezählt, bis das Programm fertig geladen ist, und die Meldung ...

```
Ready
```

... am Bildschirm erscheint.

Tippen Sie dann ein:

```
run [ENTER]
```

... und das soeben geladene Programm läuft ab. Falls ein Programm schon im Speicher war, wird es gelöscht und das neue Programm übernimmt seine Stelle.

Ein Programm kann direkt gestartet werden, indem Sie ...

```
RUN "" [ENTER]
```

... eintippen. Der Computer antwortet mit:

```
Press PLAY then any key:
```

Nachdem Sie die [PLAY] Taste und anschließend eine beliebige Taste gedrückt haben, sucht der Computer das Programm, lädt es und startet es ohne weitere Anweisung von der Tastatur. Sie können diesen Vorgang jederzeit beenden, indem Sie wie üblich auf die [ESC] Taste drücken.

1.7 Laden vorbespielter Software-Cassetten

Sie können die bisherigen Anweisungen anwenden, um eines der vielen Programme zu laden, die es für den CPC464 gibt.

Es ist jedoch ratsam, die Ladeanweisungen zu befolgen, die mit jedem Softwareprodukt mitgeliefert werden.

1.8 SAVE

Ein Programm kann gesichert (aufgenommen) werden, um es später wieder zur Verfügung zu haben. Legen Sie eine Cassette in der richtigen Weise in den Datacorder ein und schließen sie den Deckel (die Sicherungsstreifen an der Cassette dürfen nicht fehlen). Drücken sie `[REW]` um die Cassette zum Anfang zurückzuspulen, und drücken Sie sofort auf `[STOP/EJECT]` sobald die Cassette anhält. Tippen Sie ein:

`save "<`*programmtitel*`>" [ENTER]`

Der Computer antwortet mit:

`Press REC and PLAY then any key:`

Drücken Sie jetzt fest auf die `[REC]` und `[PLAY]` Tasten am Datacorder bis sie einrasten, und dann eine beliebige Taste (Buchstabe, Ziffer, Leer, oder `[ENTER]`).

Der Computer zeigt am Bildschirm:

`Saving <`*programmtitel*`> block 1`

Nachdem das Programm gesichert ist, hört die Cassette auf sich zu drehen, und das Wort: `Ready` erscheint am Bildschirm. Drücken Sie nun die `[STOP/EJECT]` Taste am Datacorder und das Programm ist gesichert.

Beachten Sie, daß Sie vorbespielte Software und Spiele nicht auf eigene Cassetten sichern können.

Solche Programme sind gegen unbefugtes Kopieren geschützt.

Grundlagen 2: DIE TASTATUR

Wir erklären jetzt die Funktionen einiger Tasten Ihres Computers. Falls Sie schon mit Computern gearbeitet haben, können Sie diesen Abschnitt übergehen.

[ENTER]

Es gibt zwei [ENTER] Tasten. Beide Tasten geben die Information, die Sie eingetippt haben, weiter an den Computer. Nachdem die [ENTER] Taste gedrückt ist, wird eine neue Zeile am Bildschirm begonnen. Jede Anweisung, die Sie eintippen, sollte mit [ENTER] abgeschlossen werden.

[DEL]

Diese Taste wird benutzt, um am Bildschirm links vom Cursor ein Zeichen (z. B. einen Buchstaben oder eine Ziffer) zu löschen, die Sie nicht brauchen.

Tippen Sie a b c d ein, und Sie sehen, daß der Buchstabe d links vom Cursor positioniert ist. Wenn Sie jetzt den Buchstaben d nicht wollen, drücken Sie [DEL] und Sie sehen wie das d entfernt wird. Wenn Sie [DEL] drücken und die Taste gedrückt halten, so werden auch die Buchstaben a b c entfernt.

[SHIFT]

Zwei [SHIFT] Tasten sind vorhanden. Wenn Sie eine dieser Tasten drücken und unten halten, während Sie ein Zeichen eingeben, wird ein Großbuchstabe oder das obere Symbol auf der Taste am Bildschirm angezeigt.

Tippen Sie den Buchstaben a ein, dann halten Sie die [SHIFT] Taste gedrückt und tippen Sie das a noch einmal. Am Bildschirm sehen Sie:

aA

Jetzt tippen Sie einige Leerstellen ein, indem Sie die Leertaste gedrückt halten. Benutzen Sie jetzt die oberste Tastenreihe mit Nummern. Tippen Sie die Ziffer 2 ein, dann halten Sie die [SHIFT] Taste gedrückt und tippen Sie die Ziffer 2 noch einmal. Am Bildschirm sehen Sie:

2"

Sie sehen nun was passiert, wenn die [SHIFT] Taste gedrückt wird, und gleichzeitig eine zweite Taste gedrückt wird. Probieren Sie dies auch mit anderen Tasten aus.

[CAPS LOCK]

Diese Taste hat eine ähnliche Funktion wie die [SHIFT] Taste, jedoch braucht diese Taste nur einmal gedrückt zu werden. Ab dann werden alle eingetippten Buchstaben als Großbuchstaben ausgegeben, die Nummerntasten werden allerdings nicht als Symbole abgebildet. Drücken Sie auf [CAPS LOCK] und tippen Sie ein:

abcdef123456

Sie werden am Bildschirm merken, wie die Buchstaben in Großbuchstaben umgewandelt wurden, die Nummern bleiben unverändert. Falls einzelne Symbole nötig sind während die [CAPS LOCK] Funktion aktiv ist, brauchen Sie nur auf die [SHIFT] Taste zu drücken wenn die gewünschte Symboltaste gedrückt wird. Tippen Sie die folgenden Tasten ein während Sie die [SHIFT] Taste gedrückt halten:

ABCDEF123456

Am Bildschirm sehen Sie:

ABCDEF!#$%&

Wenn Sie jetzt zu Kleinbuchstaben zurückkehren möchten, brauchen Sie nur noch einmal auf die [CAPS LOCK] Taste zu drücken.

Falls Sie Großbuchstaben und Symbole eintippen wollen, ohne ständig auf die [SHIFT] Taste zu drücken, dann halten Sie die [CTRL] Taste gedrückt und tippen Sie auf die [CAPS LOCK] Taste. Geben Sie ein:

abcdef123456

Am Bildschirm sehen Sie:

ABCDEF!#$%&

Während die [CTRL] und [CAPS LOCK] Tasten aktiv sind, können Sie weiterhin Ziffern eintippen, indem Sie das Zifferntastenfeld rechts vom Haupttastenfeld benutzen.

Wenn Sie jetzt die [CTRL] Taste festhalten und die [CAPS LOCK] Taste drücken, kehren Sie zu der Verarbeitungsart zurück, in der Sie vorher waren – entweder CAPS LOCK oder Kleinbuchstaben. Sie kommen wieder zu Kleinbuchstaben von CAPS LOCK, indem Sie noch einmal auf die [CAPS LOCK] Taste drücken.

[CLR]

Die Taste löscht ein Zeichen innerhalb des Cursors.

Tippen Sie ABCDEFGH ein. Der Cursor steht rechts vom letzten eingetippten Zeichen (H). Drücken Sie jetzt auf die Cursorlinkstaste [←] vier Mal. Der Cursor steht auf dem E.

Beobachten Sie, wie der Buchstabe E innerhalb des Cursors noch sichtbar ist. Drücken Sie die [CLR] Taste einmal, und sehen Sie wie das E verschwindet, und die Buchstaben FGH jeweils eine Stelle nach links verschoben werden. Der Buchstabe F ist jetzt innerhalb des Cursors. Drücken Sie noch einmal auf die [CLR] Taste und halten Sie sie fest. Sehen Sie wie das F verschwindet, gefolgt vom G und H.

[ESC]

Die [ESC] Taste wird benutzt, um aus einer Funktion auszubrechen (escape). Drücken Sie einmal auf die [ESC] Taste und der Computer unterbricht seine laufende Funktion, er arbeitet jedoch weiter, sobald eine andere Taste gedrückt wird.

Zweimaliges auf die [ESC] Taste drücken verursacht, daß der Computer seine Funktion verläßt. Der Computer erwartet jetzt neue Anweisungen von Ihnen.

Jetzt drücken Sie zweimal auf die [ESC] Taste.

WICHTIG

Falls Sie den rechten Rand erreichen, indem Sie mehr als 40 Zeichen eintippen, erscheint das nächste Zeichen auf der nächsten Zeile am linken Rand. Sie sollen NICHT auf die [ENTER] Taste drücken, wie Sie auf einer Schreibmaschine den Zeilenvorschub am rechten Zeilenende betätigen würden.

Der Computer fügt diese Funktion automatisch für Sie ein, und wird auf ein unerwartetes [ENTER] mit einer Fehlermeldung – normalerweise SYNTAX ERROR – entweder sofort oder beim Programmablauf reagieren.

SYNTAX ERROR (Syntaxfehler)

Die Meldung Syntax Error erscheint am Bildschirm, wenn der Computer nicht versteht, was Sie eingetippt haben. Tippen Sie z. B. ein:

```
printt [ENTER]
```

Am Bildschirm erscheint:

```
Syntax error
```

Die Meldung erscheint, weil der Computer printt nicht versteht.

Wenn Sie eine fehlerhafte Programmzeile eintippen, wie z. B.:

```
10 printt "abc" [ENTER]
```

erscheint die Meldung SYNTAX ERROR erst, wenn der Computer diese Anweisung verarbeitet, also dann, wenn das Programm läuft.

Tippen Sie ein:

```
run [ENTER]
```

Am Bildschirm erscheint:

```
Syntax error in 10
10 printt "abc"
```

Diese Meldung sagt Ihnen, in welcher Zeile der Fehler aufgetreten ist und zeigt Ihnen die Programmzeile mit Korrekturcursor, sodaß Sie den Fehler korrigieren können.

Drücken Sie auf die Cursorrechtstaste [→] bis der Cursor auf dem t in printt steht. Jetzt drücken Sie auf die [CLR] Taste, um das ungewollte t zu entfernen, und auf die [ENTER] Taste um die korrigierte Zeile wieder in den Computer einzugeben.

Tippen Sie run [ENTER] ein, und Sie sehen, daß der Computer die Anweisung jetzt akzeptiert und abc am Bildschirm ausgibt.

Eine Einführung in AMSTRAD BASIC

Sie finden im Abschnitt 8 eine ausführliche Beschreibung aller BASIC-Befehle, die im AMSTRAD BASIC vorkommen. In diesem Kapitel wollen wir einige der gebräuchlichsten BASIC-Befehle vorstellen.

CLS

Tippen Sie `cls` ein (clear screen: Bildschirm löschen). Sie können entweder Groß- oder Kleinbuchstaben verwenden. Dann drücken Sie auf `[ENTER]`. Sie merken, daß der Bildschirm gelöscht wird, und das Wort `Ready` und der Cursor oben links erscheint neu.

PRINT

Diese Anweisung wird benutzt, um Zeichen, Wörter oder Figuren im Programm auf dem Bildschirm anzuzeigen. Tippen Sie die folgende Anweisung ein:

```
print "hallo" [ENTER]
```

Am Bildschirm erscheint dann:

```
hallo
```

Die Anführungszeichen `" "` werden benutzt, um dem Computer mitzuteilen, was ausgegeben werden soll. `hallo` erschien am Bildschirm, sobald `[ENTER]` gedrückt wurde. Tippen Sie `cls [ENTER]`, um den Bildschirm wieder zu löschen.

RUN

Das eben gezeigte Programmbeispiel enthielt nur eine Zeile. Die meisten Programme bestehen aus vielen Programmzeilen. Vor jeder Zeile wird eine Nummer eingetippt. Diese Nummern sagen dem Computer, in welcher Reihenfolge das Programm ablaufen soll. Wenn `[ENTER]` gedrückt wird, wird das Programm im Speicher gehalten, bis es abläuft. Tippen Sie ein:

```
10 print "hallo" [ENTER]
```

Sie merken, daß `hallo` nicht auf dem Bildschirm erschien, obwohl `[ENTER]` gedrückt wurde. Hierfür muß das Wort `RUN` eingetippt werden. Tippen Sie ein:

```
run [ENTER]
```

Sie sehen jetzt `hallo` auf dem Bildschirm. Anstelle von `PRINT` können Sie das `?` benutzen, z. B.:

```
10 ? "hallo" [ENTER]
```

LIST

Es ist möglich, ein Programm anzuschauen, nachdem es eingetippt und zwischengespeichert wurde. Tippen Sie ein:

```
list [ENTER]
```

und am Bildschirm erscheint:

```
10 PRINT "hallo"
```

Es ist das Programm, das im Speicher steht.

Beachten Sie, daß das Wort `PRINT` jetzt in Großbuchstaben erscheint. Das bedeutet, daß derComputer `PRINT` als BASIC Schlüsselwort erkannt hat.

Tippen Sie `cls [ENTER]` ein, um den Bildschirm zu löschen. Dabei wird Ihr Programm nicht aus dem Speicher gelöscht, auch wenn es nicht mehr am Bildschirm zu sehen ist.

GOTO

Der `GOTO` Befehl sagt dem Computer, daß er von einer Zeile zu einer anderen springen soll, entweder um einige Zahlen zu überspringen, oder um eine Schleife zu bilden. Tippen Sie ein:

```
10 print "hallo" [ENTER]
20 goto 10 [ENTER]
```

Tippen Sie dann ein:

```
run [ENTER]
```

und Sie sehen, wie `hallo` wiederholt auf dem Bildschirm, Zeile für Zeile auf der linken Seite, erscheint. Sie können das Programm anhalten, indem Sie einmal `[ESC]` drücken.

Wenn Sie jetzt irgendeine andere Taste drücken, läuft das Programm weiter. Das Programm kann abgebrochen werden, indem Sie zweimal auf `[ESC]` drücken. Tippen Sie ein:

```
cls [ENTER]
```

um den Bildschirm zu löschen.

Sie können sehen, wie `hallo` hintereinander auf jeder Zeile den ganzen Bildschirm füllt, wenn Sie das Programm von der vorigen Seite eintippen, jedoch mit einem Strichpunkt (Semicolon) `;` nach dem Anführungszeichen`"`. Tippen Sie ein:

```
10 print "hallo"; [ENTER]
20 goto 10 [ENTER]
run [ENTER]
```

Der Strichpunkt `;` sagt dem Computer, daß er das nächste Zeichen bzw. eine Gruppe von Zeichen unmittelbar hinter das vorangehende schreiben soll. Sie können das Programm abbrechen, wenn Sie zweimal `[ESC]` drücken. Geben Sie jetzt nochmal Zeile 10 ein, dieses Mal aber mit einem Komma `,` anstelle des Strichpunktes `;`

```
10 print "hallo", [ENTER]
run [ENTER]
```

Sie werden feststellen, daß das Komma , dem Computer sagt, das nächste Zeichen (oder eine Zeichengruppe) im Abstand von 13 Spalten vom Beginn der ersten Zeichengruppe zu schreiben. Diese Einrichtung ist sehr hilfreich, wenn man Informationen in verschiedenen Spalten ausgeben will.

Beachten Sie jedoch dabei, daß das nächste Zeichen um weitere 13 Spalten verschoben wird, wenn die Anzahl der Zeichen einer Gruppe 12 übersteigt.

Drücken Sie wieder zweimal auf [ESC] um das Programm abzubrechen. Um den Computerspeicher komplett zu löschen, halten Sie nacheinander die [SHIFT], [CTRL] und [ESC] Taste gedrückt, der Computer wird zurückgesetzt.

INPUT

Mit diesem Kommando sagen Sie dem Computer, daß eine Eingabe auf ihn wartet, z. B. die Antwort auf eine Frage. Tippen Sie ein:

```
10 input "Wie alt sind Sie"; alter [ENTER]
20 print "Sie sehen juenger aus";
   alter "Jahre aus." [ENTER]
run [ENTER]
```

Am Bildschirm sehen Sie:

```
Wie alt sind Sie?
```

Geben Sie Ihr Alter ein, dann [ENTER]. Wenn Sie 18 sind, erscheint

```
Sie sehen juenger als 18 Jahre aus.
```

Dieses Beispiel zeigt, wie man das input Kommando und eine numerische Variable anwendet. Das Wort alter wurde am Ende von Zeile 10 in den Speicher gebracht, sodaß der Computer das Wort alter mit der eingegebenen Zahl in Verbindung bringen konnte. Obwohl wir den Ausdruck alter für das Alter benutzt haben, hätten wir genauso gut den Buchstaben b in unserem Beispiel benutzen können.

Setzen Sie den Computer zurück, um den Speicher zu löschen (mit den [CTRL], [SHIFT] und [ESC] Tasten). Falls Sie als Eingabe (input) Buchstaben, bzw. Buchstaben und Ziffern gewünscht hätten, so hätten Sie am Ende der Variablen das Dollarzeichen $ anfügen müssen.

Tippen Sie das folgende Programm ein: (Achten Sie darauf, in Zeile 20 nach dem o in hallo und vor dem i in ich eine Leerstelle einzugeben).

```
10 input "Wie heissen Sie"; name$ [ENTER]
20 print "hallo"; name$ ich heisse Arnold" [ENTER]
run [ENTER]
```

Am Bildschirm sehen Sie:

```
Wie heissen Sie?
```

Geben Sie Ihren Namen ein: (z. B. Franz)

Am Bildschirm erscheint:

```
hallo Franz ich heisse Arnold
```

(Vielleicht interessiert es Sie, daß *Arnold* der Codename für den Amstrad CPC464 während seiner Entwicklung war.)

Oben haben wir `name$` für die Zeichenvariable Name benutzt, obwohl es genau so gut mit einem Buchstaben, z.B. `a$`, gegangen wäre. Wir wenden jetzt die letzten beiden Beispiele in einem Programm an.

Setzen Sie den Computer zurück mit den [CTRL], [SHIFT] und [ESC] Tasten, und tippen Sie ein:

```
5 cls [ENTER]
10 input "Wie heissen Sie"; a$ [ENTER]
20 "Wie alt sind Sie"; b [ENTER]
30 print "Ich muss sagen";a$ " Sie sehen nicht wie";
   b "Jahre aus" [ENTER]
run [ENTER]
```

Wir haben zwei Variable in diesem Programm benutzt, `a$` für den Namen und `b` für das Alter. Am Bildschirm sehen Sie:

```
Wie heissen Sie?
```

Geben Sie Ihren Namen (z. B. Franz) und [ENTER] ein, dann werden Sie gefragt:

```
Wie alt sind Sie?
```

Geben Sie Ihr Alter (z. B. 18) und [ENTER] ein. Wenn Sie Franz heißen und 18 sind, sehen Sie jetzt am Bildschirm:

```
Ich muss sagen Franz Sie sehen nicht wie 18 Jahre aus
```

EDITIEREN EINES PROGRAMMS

Falsch eingegebene Programmzeilen, die Syntaxfehler oder sonstige Fehler verursacht haben, können korrigiert (editiert) werden, anstatt sie neu einzutippen. Um zu zeigen, wie so etwas geht, tippen Sie das vorige Programm folgendermaßen fehlerhaft ein:

```
5 clss [ENTER]
10 input "Wie heissen Sie"; a$ [ENTER]
20 input "Wie alt sin Sie"; b [ENTER]
30 print "Ich muss sagen ";a$ "Sie sehen nicht wie";
   b "Jahre aus" [ENTER]
```

Wir haben 3 Fehler im Programm:

In Zeile 5 haben wir `clss` anstatt `cls` eingetippt.
In Zeile 20 haben wir `sin` anstatt `sind` eingetippt.
In Zeile 30 haben wir die Leerstelle zwischen `sagen` und " vergessen.

Wir haben 3 Möglichkeiten, ein Programm zu editieren. Die erste ist, die Zeile komplett neu einzutippen. Wenn eine Zeile eingetippt wird, ersetzt sie eine ggf. vorhandene Zeile im Speicher mit der gleichen Nummer. Die zweite Möglichkeit ist die EDIT Methode und drittens haben wir die sog. *COPY Cursor* Methode.

EDIT-Methode

Um den Fehler in Zeile 5 zu korrigieren, tippen Sie ein:

```
edit 5 [ENTER]
```

Zeile 5 wird ausgedruckt mit dem Cursor auf der `5`. Drücken Sie die Cursorrechtstaste [→], √ bis der Cursor auf dem zusätzlichen `s` in `clss` steht, und drücken Sie jetzt auf die [CLR] Taste. Das `s` verschwindet. Drücken Sie jetzt [ENTER] und Zeile `5` wird im Speicher korrigiert.

Tippen Sie ein:

```
list [ENTER]
```

um die Korrektur zu überprüfen.

COPY Cursor-Methode

Um die Fehler in den Zeilen 20 und 30 zu korrigieren, drücken Sie gleichzeitig auf die [SHIFT] Taste und auf die Cursorobentaste [↑], bis der COPY-Cursor auf den Anfang von Zeile 10 zeigt. Der Hauptcursor bewegt sich nicht, sodaß jetzt zwei Cursor auf dem Bildschirm sind. Drücken Sie dann auf die [COPY] Taste, bis der COPY-Cursor auf der Leerstelle zwischen sin und Sie steht. Sie merken, daß Zeile 20 nochmal auf der untersten Zeile geschrieben wird und der Hauptcursor an der gleichen Stelle hält wie der COPY-Cursor. Tippen Sie d ein. Dies wird nur auf der untersten Zeile eingefügt.

Der Hauptcursor hat sich bewegt, aber der COPY-Cursor ist stehengeblieben. Drücken Sie jetzt auf die [COPY] Taste bis die Zeile 20 vollständig geschrieben ist. Drücken Sie die [ENTER] Taste und diese neue Zeile 20 wird in den Speicher übernommen. Der COPY-Cursor verschwindet und der Hauptcursor steht jetzt unter der neuen Zeile 20.

Der verbleibende Fehler kann in ähnlicher Weise korrigiert werden. Drücken Sie wieder gleichzeitig auf die [SHIFT] Taste und auf die Cursorobertaste [↑] bis der COPY-Cursor auf den Anfang von Zeile 30 zeigt.

Drücken Sie die [COPY] Taste bis der COPY-Cursor über dem Anführungszeichen " neben sagen steht. Drücken Sie einmal auf die Leerstelle. Eine Leertaste wird in der untersten Zeile eingefügt. Drücken Sie erneut die [COPY] Taste bis die Zeile 30 komplett übernommen ist. Dann drücken Sie auf [ENTER]. Sie können das ganze Programm überprüfen, indem Sie list [ENTER] eintippen.

Setzen Sie den Computer mit [CTRL], [SHIFT] und [ESC] zurück.

IF THEN

Wir erweitern jetzt unser Programm, indem wir die neuen Kommandos IF und THEN einbauen.

Tippen Sie folgendes ein: Beachten Sie, daß wir zwei neue Symbole eingefügt haben, < bedeutet *kleiner als* und ist neben der M-Taste, > bedeutet *größer als* und steht neben der <-Taste (kleiner als).

```
5 cls [ENTER]
10 input "Wie heissen Sie"; a$ [ENTER]
20 input "Wie alt sind Sie"; alter [ENTER]
30 if alter < 13 then 60 [ENTER]
40 if alter < 20 then 70 [ENTER]
50 if alter > 19 then 80 [ENTER]
60 print "Sag mal " a$ "Du bist mit " alter
   "Jahren noch kein Teenager":end [ENTER]
70 print "Also " a$ " ist ein Teenager mit "
   alter "Jahren":end [ENTER]
80 print "Sagen Sie " a$ " mit " alter "sind
   "Sie kein Teenager mehr" [ENTER]
```

Überprüfen Sie Ihr Programm mit:

```
list [ENTER]
```

Jetzt tippen Sie ein:

```
run [ENTER]
```

Beantworten Sie jetzt alle Fragen des Computers, und sehen Sie, was passiert.

Sie sehen jetzt, welche Funktionen `IF` und `THEN` im Programm haben. Wir haben auch das Wort `end` am Ende der Zeilen 60 und 70 hinzugefügt. Das Schlüsselwort `end` bedeutet wortwörtlich ‚beende das Programm'. Ohne `end` in Zeile 60 würde das Programm weiter laufen und die Zeilen 70 und 80 bearbeiten. Ebenso würde ohne `end` am Ende von Zeile 70 das Programm weiterlaufen und Zeile 80 ausführen. Der Doppelpunkt `:` erlaubt es Ihnen zwei oder mehr Anweisungen in einer Programmzeile zu schreiben. Der Doppelpunkt `:` vor dem Schlüsselwort `end` trennt es von der vorangegangenen Anweisung. Wir haben auch eine Zeile 5 eingebaut, um den Bildschirm am Programmanfang zu löschen. Wir werden dies von jetzt an in den folgenden Programmen immer tun, damit unsere Arbeit sauberer aussieht.

Andere Schlüsselwörter in Verbindung mit `IF` und `THEN` sind u. a. `ELSE`, `OR` und `GOTO`.

Setzen Sie den Computer mit `[CTRL]`, `[SHIFT]` und `[ESC]` zurück.

FOR TO NEXT

Wir zeigen jetzt die Anwendung der `FOR`, `TO` und `NEXT` Kommandos. In einem Beispiel zeigen wir, wie der Computer das 12-er Einmaleins (1x12, 2x12, 3x12, usw.) ausdrucken kann. Im Beispiel bedeutet `*` multiplizieren.

```
5 cls [ENTER]
10 for a = 1 to 20 [ENTER]
20 print a"* 12 ="a*12 [ENTER]
30 next a [ENTER]
run [ENTER]
```

Sie merken, daß die Spalten unsauber aussehen. Tippen Sie deshalb:

```
5 cls [ENTER]
10 for a = 1 to 9 [ENTER]
20 print a" * 12 ="a*12 [ENTER]
30 next a [ENTER]
40 for a = 10 to 20 [ENTER]
50 print a"* 12 ="a*12 [ENTER]
60 next a [ENTER]
run [ENTER]
```

Probieren Sie dieses Programm mit anderen Zahlen, z. B. mit der 17. Dafür müssen Sie die 12 in den Zeilen 20 und 50 durch 17 ersetzen. Setzen Sie zum Schluß den Computer mit `[CTRL]`, `[SHIFT]` und `[ESC]` zurück. Sie können auch die Schrittweite im `FOR TO NEXT` Kommando mit `STEP` spezifizieren. Weitere Information hierzu unter `FOR` im Kapitel 8.

EINFACHE ARITHMETIK

Ihr CPC464 kann sehr einfach als Tischrechner eingesetzt werden.

Die folgenden Beispiele erläutern das näher. Wir verwenden in diesem Abschnitt das ? anstatt PRINT zu schreiben. Die Antwort wird ausgegeben, so bald die [ENTER] Taste gedrückt wird.

ADDITION

(drücken Sie für plus die [SHIFT] und ; Tasten)

Tippen Sie:

```
?3+3 [ENTER]
 6
```

(das Gleichheitszeichen = wird nicht eingegeben)

Tippen Sie:

```
?8+4 [ENTER]
 12
```

SUBTRAKTION

(benutzen Sie die = Taste ohne Shift für Minus)

Tippen Sie:

```
?4-3 [ENTER]
 1
```

Tippen Sie:

```
?8-4 [ENTER]
 4
```

MULTIPLIKATION

(benutzen Sie die [SHIFT] und die : Taste zum Multiplizieren, * bedeutet x)

Tippen Sie:

```
?3*3 [ENTER]
 9
```

Tippen Sie:

```
?8*4 [ENTER]
 32
```

DIVISION

(benutzen Sie die ? Taste ohne Shift zum Dividieren, / bedeutet ÷)

Tippen Sie:

```
?8/4 [ENTER]
 2
```

QUADRATWURZEL

Sie errechnen die Quadratwurzel einer Zahl, indem Sie `sqr( )` eintippen. Die Zahl, deren Quadratwurzel Sie möchten, muß in Klammern stehen.

Tippen Sie:

```
?sqr(16) [ENTER]
 4
```
(das bedeutet $\sqrt{16}$)

Tippen Sie:

```
?sqr (100) [ENTER]
 10
```

POTENZEN

(benutzen Sie die £ Taste ohne Shift zum Potenzieren)

Das Produkt aus n gleichen Faktoren b wird als Potenz von b bezeichnet
z. B. 3 Quadrat (3^2), 3 Kubik (3^3), usw., tippen Sie:

```
?3↑3 [ENTER]
 27
```
(das bedeutet 3^3)

Tippen Sie:

```
?8↑4 [ENTER]
 4096
```
(das bedeutet 8^4)

KUBIKWURZEL

Die Kubikwurzel wird ähnlich berechnet wie die Quadratwurzel.

Sie finden die Kubikwurzel von 27 ($\sqrt[3]{27}$) indem Sie tippen:

```
?27↑(1/3) [ENTER]
 3
```

Für die Kubikwurzel von 125 tippen Sie:

```
?125↑(1/3) [ENTER]
 5
```

GEMISCHTE KALKULATIONEN

Gemischte Kalkulationen werden vom Computer verstanden, werden jedoch unter Berücksichtigung einiger Prioritäten berechnet.

Höchste Priorität hat die Multiplikation, danach die Division, dann die Addition und letztlich die Subtraktion. Diese Prioritäten gelten jedoch nur, wenn nur diese vier Rechnungsarten vorhanden sind. Die Prioritäten werden später einschließlich Potenzen usw. weiter erläutert.

Nehmen wir folgende Berechnung

```
3+7-2*7/4
```

Vielleicht denken Sie, daß das so gerechnet wird:

```
3+7-2  *7  /4
=    8 *7  4
=       56  /4
=       14
```

in Wirklichkeit wird folgendermaßen gerechnet:

```
3+7-2*7/4
=3+7-14/4
=3+7-3,5
=10-3,5
=6.5
```

Sie können dies nachvollziehen, indem Sie folgendes eintippen:

```
?3+7-2*7/4 [ENTER]
 6.5
```

Die Reihenfolge der Berechnung kann verändert werden, wenn Sie Klammern verwenden. Der Ausdruck in Klammern hat Vorrang vor der Multiplikation usw. außerhalb der Klammern. Überzeugen Sie sich durch folgende Rechnung mit Klammern:

```
?(3+7-2)*7/4 [ENTER]
 14
```

WEITERE POTENZEN

Falls Sie sehr große oder sehr kleine Zahlen brauchen, ist eine andere Schreibweise vorteilhafter. Der Buchstabe E wird als Potenz zur Basis 10 verstanden. Sie dürfen sowohl das große E als auch das kleine e verwenden.

Die Zahl 300 kann z. B. auch als $3x10^2$ geschrieben werden. In der wissenschaftlichen Schreibweise schreibt man diese 3E2. Ähnlich ist es mit 0,03 (auch $3x10^{-2}$). In der neuen Schreibweise haben wir 3E−2. Probieren Sie folgende Beispiele aus:

1. 30x10

Sie können eintippen:

```
?30*10 [ENTER]
 300
```

oder Sie können eintippen:

```
?3e1*1e1 [ENTER]
 300
```

2. 3000x1000

Tippen Sie:

```
?3e3*1e3 [ENTER]
 3000000
```

3. 3000x0,001

Tippen Sie:

```
?3e3*1e-3 [-ENTER]
 3
```

Grundlagen 3: Graphik, Darstellungsarten und Musik

Auf dem Bildschirm des CPC464 Colour Personal Computer sind drei verschiedene Darstellungsarten (Modi) möglich: Modus 0, Modus 1 und Modus 2.

Nach dem Einschalten des Computers ist automatisch Modus 1 eingestellt.

Um die einzelnen Darstellungsarten zu verdeutlichen, drücken Sie auf die Taste mit der Ziffer `1`. Halten Sie die Taste fest, bis zwei Zeilen mit 1-ern voll sind. Wenn Sie jetzt zählen, sehen Sie, daß auf jeder Zeile 40 Ziffern sind. Das heißt, daß im Modus 1 vierzig Spalten zur Verfügung stehen. Drücken Sie jetzt auf `[ENTER]` – Sie bekommen zwar einen `Syntaxfehler`, aber der stört uns nicht. Wir kommen nur schneller auf `Ready` zurük, so daß wir die nächste Anweisung eingeben können.

Tippen Sie: `mode 0 [ENTER]`

Sie sehen, daß die Zeichen auf dem Bildschirm jetzt größer sind. Drücken Sie wieder auf die 1-er Taste bis zwei Zeilen vollgeschrieben sind. Jetzt sind nur noch 20 Ziffern in einer Zeile. Das heißt, daß wir in Mode 0 nur 20 Spalten zur Verfügung haben. Drücken Sie erneut auf `[ENTER]`

Tippen Sie jetzt: `mode 2 [ENTER]`

Dies ist der kleinste Modus, und wenn Sie jetzt eine Reihe von 1-ern eingeben, dann sehen Sie 80 Ziffern. In Mode 2 haben wir 80 Spalten. Nochmal:

Mode 0 = 20 Spalten
Mode 1 = 40 Spalten
Mode 2 = 80 Spalten

Drücken Sie jetzt noch einmal auf `[ENTER]`

FARBEN

Wir können unter 27 Farben auswählen. Diese erscheinen auf einem grünen Bildschirm (GT65) als unterschiedlich helle Grünstufen. Wenn Sie den Monitor GT65 besitzen, können Sie einen Amstrad MP2 Modulator/Stromversorgung dazukaufen um die Farbmöglichkeiten Ihres Computers auf Ihrem eigenen Fernsehgerät einzusetzen.

Im Modus 0 können bis zu 16 der insgesamt 27 Farben gleichzeitig verwendet werden.
Im Modus 1 können bis zu 4 der 27 Farben gleichzeitig verwendet werden.
Im Modus 2 können bis zu 2 der 27 Farben gleichzeitig verwendet werden.

Sie können die Farben des Bildrandes `(BORDER)`, des Papiers `(PAPER)` – die Fläche auf der die Zeichen stehen – oder des Schreibstiftes `(PEN)` – das Zeichen selbst – unabhängig voneinander verändern.

Die Palette der 27 möglichen Farben ist mit der jeweils zugehörigen Tinten-`(INK)`-nummer in Tabelle 1 aufgeführt.

FARBTABELLE

Tinte (INK)	Farbe	Tinte (INK)	Farbe
0	Schwarz	14	Pastellblau
1	Blau	15	Orange
2	Hellblau	16	Rosa
3	Rot	17	Pastellmagenta
4	Magenta	18	Hellgrün
5	Hellviolett	19	Seegrün
6	Hellrot	20	helles Blaugrün
7	Purpur	21	Limonengrün
8	helles Magenta	22	Pastellgrün
9	Grün	23	Pastellblaugrün
10	Blaugrün	24	Hellgelb
11	Himmelblau	25	Pastellgelb
12	Gelb	26	Leuchtendweiß
13	Weiß		

Wie vorher erklärt, befindet sich der Computer

Wie vorher erklärt, befindet sich der Computer nach dem Einschalten im Modus 1. Um auf Modus 1 zurückzukommen, tippen Sie:

```
Mode 1 [ENTER]
```

Die Farben von `BORDER`, `PAPER` und `PEN` beim Einschalten sind:

BORDER (Rand) : Farbnummer 1 (Blau)
PAPER (Bildschirm) : Farbnummer 1 (Blau)
PEN (Zeichen) : Farbnummer 24 (Hellgelb)

Der Rand (BORDER) ist die Zone um die eigentliche Schreibfläche (PAPER). Nach dem Einschalten sind sowohl BORDER als auch PAPER blau. PAPER (Papier) ist die eigentliche Schreibfläche innerhalb des Randes (BORDER), in der die Zeichen stehen können. PEN ist die Farbe der Zeichen.

Dies wird vielleicht klarer, wenn wir PEN (Schreibstift) und PAPER (Papier) des Bildschirms mit einem Kugelschreiber und einem Notizblock vergleichen. So wie wir die Farbe der Tinte (INK) in einem Schreibstift (PEN) austauschen können, so können wir die Farbe der Zeichen am Bildschirm auswechseln. Das Papier unseres Notizblocks kann auch verschiedenfarbig sein, genauso ist es mit der Farbe der Schreibfläche (PAPER) unseres Bildschirms.

Die Randfarbe (BORDER) verändern Sie, indem Sie eintippen:

```
border 0 [ENTER]
```

Die Randfarbe wechselt von blau auf schwarz. In Tabelle 1 sehen Sie, daß 0 für Schwarz steht. Sie können die Randfarbe in jede dieser Farben ändern, indem Sie border und dann die Farbnummer eintippen.

Tippen Sie jetzt:

```
cls [ENTER]
```

um den Bildschirm zu löschen.

Wenn Sie die Papierfarbe (PAPER) verändern wollen, geben Sie ein:

```
paper 2 [ENTER]
```

die Hintergrundfarbe hinter dem Wort ready ist jetzt ein helles Blaugrün.

Tippen Sie jetzt:

```
cls [ENTER]
```

um den Bildschirm mit der neuen PAPERfarbe zu füllen.

Sie können auch die Farbe des PEN (Schreibstift) verändern, z. B. mit:

```
pen 3 [ENTER]
```

Die Farbe des Stiftes hat sich jetzt verändert, und das Wort ready wird hellrot angezeigt.

Tippen Sie jetzt:

```
cls [ENTER]
```

Sie werden obige Vorgänge besser verstehen, wenn Sie sich auch Tabelle 2 anschauen. Nach dem Einschalten hat PAPER die Nummer 0. In Tabelle 2 finden Sie in der ersten Spalte unter PEN die Nummer 0. In der gleichen Zeile unter Modus 1 finden Sie die Farbnummer 1.

Wenn Sie sich jetzt die Farbtabelle (Tabelle 1) ansehen, finden Sie unter Nummer 1 die Farbe Blau. Diese Farbe hat das Papier (PAPER) wenn Sie den Computer einschalten.

Wir haben gerade PAPER (Papier) auf Nummer 2 gesetzt. In der linken Spalte in Tabelle 2 finden Sie Papiernummer 2; in der gleichen Zeile finden Sie unter Modus 1 die Farbnummer 20. Aus Tabelle 1 können Sie ablesen, daß 20 für helles Blaugrün steht.

	Tintenfarbe		
Paper/Pen	Mode 0	Mode 1	Mode 2
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	24	24
6	2	20	1
7	8	6	24
8	10	1	1
9	12	24	24
10	14	20	1
11	16	6	24
12	18	1	1
13	22	24	24
14	blinkend 1,24	20	1
15	blinkend 16,11	6	24

Tabelle 2: `PAPER/PEN/MODE/TINTE` Verweistabelle

Nachdem der Computer eingeschaltet ist, hat der Stift (PEN) die Nummer 1. Beim Nachschauen in Tabelle 2 sehen Sie, daß im Modus 1 für PEN die Farbnummer 24 steht. In Tabelle 1 finden Sie für 24 die Farbangabe Hellgelb. Das ist auch die Farbe der Zeichen auf dem Bildschirm nach dem Einstellen.

Wir haben vorhin den Stift (PEN) auf 3 gesetzt. In Tabelle 2 sehen Sie, daß PEN 3 der Farbe 6 im Modus 1 entspricht. Aus Tabelle 1 sehen Sie, daß Farbe 6 Hellrot ist.

Wir benutzen gerade PAPER 2 und PEN 3. Diese Farben können wir weiter verändern. Das machen wir jetzt mit dem INK Befehl. Der INK Befehl benötigt zwei Ziffern; die erste sagt, welche PEN oder PAPER Nummer zu ändern ist, die zweite gibt die neue Farbe für PEN oder PAPER an. Sehen Sie sich in Tabelle 1 die Farbnummern an. Als Beispiel ändern wir die Farbe von PAPER 2 in Schwarz und die Farbe von PEN 3 in leuchtendweiß.

Aus Tabelle 1 entnehmen Sie, daß Schwarz die Farbnummer 0 und Leuchtendweiß die Farbnummer 26 hat.

Tippen Sie:

```
ink 2,0 [ENTER]
```

(Wie gesagt, 2 ist die aktuelle PAPER Nummer und 0 ist Schwarz)

Jetzt tippen Sie:

```
ink 3,26 [ENTER]
```

(3 ist die aktuelle PEN Nummer, 26 ist Leuchtendweiß)

Setzen Sie jetzt den Computer komplett zurück, indem Sie nacheinander die [CTRL], [SHIFT] und [ESC] Tasten drücken und festhalten.

Wie schon oben erklärt, hat nach dem Einschalten des Computers oder nach dem Zurücksetzen (mit [CTRL], [SHIFT] und [ESC]) PAPER die Nummer 0 und PEN die Nummer 1. Die Farbe von PAPER ist 1 (Blau) und die Farbe des PEN ist 24 (Hellgelb). Wir würden diese als ink 0,1 für das Papier und ink 1,24 für den Stift eintippen. Um diese sofort in leuchtendweiße Zeichen (PEN) auf schwarzem Hintergrund (PAPER) zu ändern, tippen Sie:

```
ink 0,0 [ENTER]
```

und dann:

```
ink 1,26 [ENTER]
```

BLINKENDE FARBEN

Wir können die Zeichen in zwei verschiedenen Farben abwechselnd blinken lassen. Diese Funktion erreichen wir, indem wir eine zweite Farbnummer im INK Befehl für PEN angeben.

Sie sehen die Zeichen am Bildschirm abwechselnd leuchtendweiß und hellrot blinken, wenn Sie folgenden Befehl eintippen:

```
ink 1,26,2 [ENTER]
```

In diesem Fall ist 1 die PEN Nummer, während 26 leuchtendweiß und 6 die Alternativfarbe hellrot bedeutet.

Wir können auch die Papierfarbe hinter den Zeichen zwischen zwei Farben blinken lassen. Tippen Sie:

```
ink 0,9,24 [ENTER]
```

Hier ist 0 die PAPER Nummer, 9 ist die Farbe grün und 24 ist die Alternativfarbe hellgelb.

Setzen Sie jetzt den Computer zurück mit [CTRL], [SHIFT] und [ESC].

Beachten Sie, daß im Modus 0 für PEN 14 und 15 bzw. PAPER 14 und 15 schon zwei blinkende Farben vorgesehen sind. Hierzu ist es nicht erforderlich, eine extra Ziffer im INK Kommando anzugeben.

Tippen Sie:

```
mode 0 [ENTER]
pen 15 [ENTER]
```

Am Bildschirm blinkt nun das Wort ready zwischen himmelblau und rosa. Tippen Sie jetzt:

```
paper 14 [ENTER]
cls [ENTER]
```

Sie sehen jetzt, daß nicht nur das Wort ready zwischen himmelblau und rosa blinkt, sondern auch die Hintergrundfarbe (PAPER) zwischen gelb und blau.

Diese Anfangsfarben können auch mit dem INK Kommando für PEN oder PAPER verändert werden. Sie ändern die Farben für PEN in blinkend schwarz und leuchtendweiß mit dem folgenden Kommando:

```
ink 15,0,26 [ENTER]
```

In diesem Fall ist 15 die PEN Nummer, 0 ist die Farbe schwarz und 26 ist die Alternativfarbe leuchtendweiß.

Zum Schluß lassen wir den Rand (BORDER) zwischen zwei Farben blinken. Wir geben im BORDER Kommando zwei Farbnummern an. Tippen Sie:

```
border 6,9 [ENTER]
```

Der Rand blinkt jetzt zwischen hellrot und grün.

Setzen Sie jetzt den Computer zurück mit [CTRL], [SHIFT] und [ESC].

VORFÜHRPROGRAMM

Um die weiteren Farben, die es gibt, zu zeigen, tippen Sie das folgende Programm ein und lassen Sie es ablaufen.

Wir haben einige Geräuscheffekte (sound) in das Programm eingebaut. Diese werden etwas später erklärt.

```
 10 mode 0: ink 0,2: ink 1,24: paper 0 [ENTER]
 20 pen 1: for b=0 to 26: border b [ENTER]
 30 locate 3,12: print "Randfarbe"; b [ENTER]
 40 sound 4,(40-b) [ENTER]
 50 for t=1 to 600: next t: next b: cls [ENTER]
 60 for p=0 to 15: paper p: pen 5: print "paper";
    p:print [ENTER]
 70 for n=0 to 15:pen n: print "pen";n [ENTER]
 80 sound 1,(n*20+p) [ENTER]
 90 for t=1 to 100: next t : next n [ENTER]
100 for t=1 to 1000: next t: cls: next p [ENTER]
110 cls: paper 0: pen 1: locate 7,12: print
    "E N D E": for t=1 to 2000: next t [ENTER]
120 mode 1: border 1: ink 0,1: ink 1,24: paper 0:
    pen 1 [ENTER]

run [ENTER]
```

GRAPHIK

Ab jetzt werden wir nicht mehr nach jeder Zeile [ENTER] schreiben. Wir gehen davon aus, daß Sie das jetzt automatisch tun.

Der Computer hat eine ganze Reihe von Symbolen intern gespeichert. Mit dem Kommando chr$() können Sie diese Symbole ausgeben lassen. In den Klammern sollte die Nummer des Symbols, eine Zahl zwischen 32 und 255 stehen.

Setzen Sie den Computer mit [CTRL], [SHIFT] und [ESC] zurück, und tippen Sie dann:

```
print chr$(250)
```

Am Bildschirm sehen Sie Symbol 250, einen Mann, der nach rechts läuft.

Das folgende Programm können Sie benutzen, um alle vorhandenen Symbole mit ihren zugehörigen Nummern auszugeben. Tippen Sie das Programm ein, vergessen Sie dabei aber nicht, nach jeder Zeile die [ENTER] Taste zu drücken.

```
10 for n=32 to 255: print n;chr$(n);
20 next n
run
```

Im Anhang III finden Sie alle Symbole mit ihren zugehörigen Referenznummern.

LOCATE (Positioniere Cursor)

Dieses Kommando bringt den Cursor an eine gewünschte Stelle auf dem Bildschirm. Ohne LOCATE Kommando fängt der Cursor in der obersten, linken Ecke des Bildschirms an. Diese hat die X-, Y-Koordinaten 1,1 (x ist die waagerechte und y ist die senkrechte Position). Im Modus 1 stehen 40 Spalten und 25 Zeilen zur Verfügung. Die Cursorposition in der Mitte der obersten Zeile hat im Modus 1 die X-, Y-Koordinaten 20,1.

Um dies zu sehen, tippen Sie (denken Sie an [ENTER]):

```
mode 1
```
...der Bildschirm wird gelöscht, der Cursor geht nach oben links

```
10 locate 20,1
20 print chr$(250)
run
```

Um sich zu überzeugen, daß dies die oberste Zeile ist, schreiben Sie:

```
border 0
```

Der Rand (BORDER) ist jetzt schwarz und der kleine Mann ist auf der obersten Bildschirmzeile zu sehen.

In Mode 0 stehen nur 20 Spalten, aber noch die gleichen 25 Zeilen zur Verfügung. Tippen Sie:

```
mode 0
run
```

Der kleine Mann steht jetzt in der rechten oberen Ecke des Bildschirms, weil die X-Koordinate 20 die letzte Spalte im Modus 0 ist.

Im `Modus 2` gibt es 80 Spalten und 25 Zeilen. Wir benutzen das gleiche Programm. Sie können sicherlich erraten, wo der kleine Mann jetzt steht.

```
mode 2
run
```

Probieren Sie jetzt selbst ein wenig, indem Sie die Nummern für `LOCATE` und `CHR$()` verändern und damit verschiedene Symbole überall auf dem Bildschirm plazieren können. Tippen Sie z. B.:

```
locate 20,12: print chr$(240)
```

Sie sehen einen Pfeil in der Bildschirmmitte. In diesem Befehl ist

20 die waagerechte (X) Koordinate (Bereich 1 – 40)
12 die senkrechte (Y) Koordinate (Bereich 1 – 25)
240 die Nummer des gezeigten Symbols (Bereich 32 – 255)

Das Symbol 250 können Sie mit dem folgenden Programm wiederholt über den ganzen Bildschirm ausgeben, tippen Sie:

```
5 cls
10 for x=1 to 39
20 locate x,20
40 print chr$(250)
50 next x
60 goto 5
run
```

Drücken Sie zweimal die `[ESC]` Taste, um das Programm abzubrechen.

Wenn das Zeichen am Bildschirm gelöscht werden soll, bevor das nächste ausgegeben wird, tippen Sie:

```
40 print " "; chr$(250)
```

(Diese neue Zeile 40 überschreibt die bisherige Zeile 40.)

Jetzt tippen Sie:

```
run
```

Um die Bewegung des Zeichens quer über den Bildschirm zu verbessern fügen Sie folgende Zeile hinzu:

```
30 call &bd19
```

Das Programm kann noch weiter verbessert werden. Wir bauen einige Verzögerungsschleifen ein und benutzen auch ein anderes Symbol für den Rücklauf.
Tippen Sie:

```
list
```

Jetzt ergänzen Sie das Programm mit den folgenden Zeilen:

```
60 for n=1 to 300: next n
65 for x=39 to 1 step-1
70 locate x,20
75 call &bd19
80 print chr$(251);" "
85 next x
90 for n=1 to 300: next n
95 goto 10
run
```

Probieren Sie auch dieses interessante kleine Programm aus. Wir haben einige neue Befehle benutzt, die wir etwas später erläutern werden. Tippen Sie jetzt einfach ein:

```
new

10 mode 1
20 locate 21,14: print chr$(244)
30 tag
40 for x=0 to 624 step 2
50 mover -16,0
60 if x<308 or x<340 then y=196: goto 90
70 if x<324 then y=x-104: goto 85
80 y=536-x
85 sound 1,0,20,7
90 move ox, oy: print " ";:ox=x: oy=y
100 move x,y
110 if (x mod 4) = 0 then print chr$(250);
    else print chr$(251);
120 for n=1 to 4: call &bd19: next n
130 next x
140 tagoff
150 goto 20
run
```

PLOT

Anders als das LOCATE Kommando benutzen wir das PLOT Kommando, um den Graphikcursor zu positionieren, dieses Mal mit Pixel-Koordination. (Ein Pixel ist ein winziges Segment des Bildschirms.

Beachten Sie, daß der Graphikcursor unsichtbar ist und nicht identisch ist mit dem normalen Cursor.

Wir haben 640 waagerechte Pixel und 400 senkrechte Pixel. Die X-, Y-Koordinaten beziehen sich auf die unterste, linke Ecke und fangen mit 0,0 an. Diese Pixel-Koordinaten bleiben im Modus 0, 1 und 2 immer gleich.

Um dies zu testen, setzen Sie den Computer mit [CTRL], [SHIFT] und [ESC] zurück, und tippen Sie dann (denken Sie an [ENTER]):

```
plot 320,200
```

Ein kleiner Punkt erscheint in der Bildschirmmitte.

Jetzt ändern Sie die Darstellungsart (mode) mit:

```
mode 0
plot 320,200
```

Der Punkt ist immer noch in der Mitte, aber er ist jetzt größer. Ändern Sie noch Mal den Modus und tippen Sie:

```
mode 2
plot 320,200
```

Der Punkt ist immer noch in der Mitte, aber viel kleiner.

Versuchen Sie jetzt einige Punkte auf den Bildschirm zu bringen, an verschiedenen Stellen, in verschiedenen Größen. Wenn Sie fertig sind, löschen Sie den Bildschirm und kehren Sie in den Modus 1 zurück:

```
mode 1
```

DRAW

Setzen Sie den Computer erst zurück [CTRL], [SHIFT] und [ESC]. Das DRAW Kommando zeichnet eine Linie ausgehend von der aktuellen Position des Grafikzeigers. Zeichnen Sie ein Viereck am Bildschirm mit folgendem Programm. Wir setzen zunächst den Grafikzeiger mit dem PLOT Kommando an die gewünschte Stelle. Dann zeichnen wir eine Linie ab dieser Stelle nach oben in Richtung der obersten, linken Ecke. Ab hier zeichnen wir eine Linie zur rechten Ecke, usw.

Tippen Sie:

```
5 cls
10 plot 10,10
20 draw 10,390
30 draw 630,390
40 draw 630,10
50 draw 10,10
60 goto 10
run
```

Drücken Sie jetzt zweimal auf [ESC] um das Programm abzubrechen.

Fügen Sie jetzt die folgenden Zeilen in das Programm ein, damit ein zweites Rechteck innerhalb des ersten gemalt wird. Tippen Sie:

```
60 plot 20,20
70 draw 20,380
80 draw 620,380
90 draw 620,20
100 draw 20,20
200 goto 10
run
```

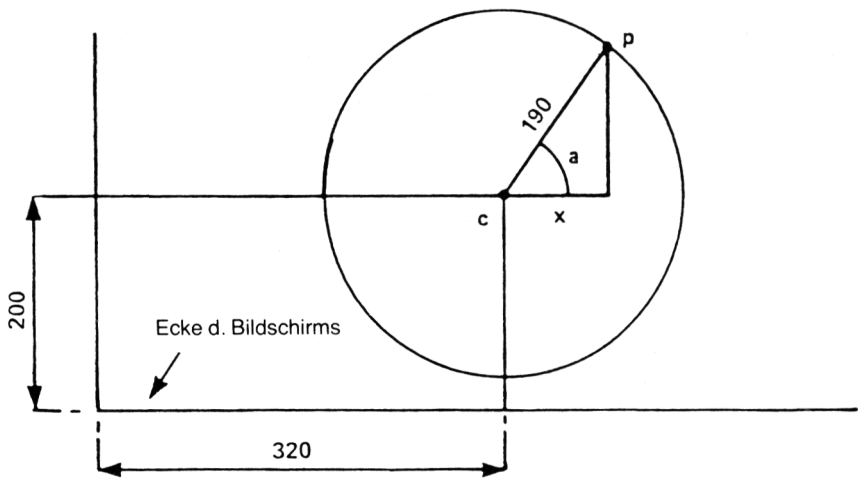
Drücken Sie zweimal auf [ESC] um das Programm abzubrechen.

KREISE

Kreise können entweder geplottet oder gezeichnet werden. Man kann einen Kreis bilden, wenn man die x, y-Koordinaten jedes Punktes auf dem Kreisumfang plottet.

Schauen Sie sich das Bild unten an und Sie entdecken, daß der Punkt p auf dem Kreisumfang mit den x- und y-Koordinaten geschrieben werden kann. Diese sind:

x = 190*cos(a) und y = 190*sin (a)



NEW

Wir haben bereits das Schlüsselwort NEW benutzt, bevor wir das eigentliche Programm eintippten. NEW sagt dem Computer, daß der Speicher zu löschen ist, ungefähr so, als wenn wir [CTRL], [SHIFT] und [ESC] drücken würden. Der Bildschirm wird jedoch nicht gelöscht, sondern nur der interne Speicher. Es ist manchmal nützlich das alte Programm vor Augen zu haben, während wir das neue Programm schreiben.

Wie man die Punkte eines Kreises plottet:

Im vorigen Programm haben wir alle Punkte bezogen auf die linke untere Ecke des Bildschirms geplottet. Wir wollen den Kreis in der Bildschirmmitte haben, also um die Koordinaten 320,200, so daß wir diese Koordinaten zu unseren errechneten Punkten addieren müssen.

Ein Programm für den Kreis könnte wie folgt aussehen. Tippen Sie:

```
new
5 cls
10 for a=1 to 360
15 deg
20 plot 320,200
30 plot 320+190*cos(a), 200+190*sin(a)
40 next
run
```

Der Radius des Kreises kann reduziert werden, wenn wir die Zahl 190 verkleinern.

Der Kreis wird anders geschrieben (im Bogenmaß), wenn wir die Zeile 15 entfernen. Löschen Sie die Zeile 15, indem Sie einfach

```
15
```

eintippen.

Ein geschlossener Kreis wird geschrieben, wenn wir das Schlüsselwort PLOT durch DRAW ersetzen. Zeile 30 wäre dann:

```
draw 320+190*cos(a), 200+190*sin(a)
```

Probieren Sie es aus, mit und ohne Zeile 15.

Sie haben in Zeile 40 gesehen, daß wir next statt next a schrieben.

Es ist erlaubt, einfach next zu schreiben. Der Computer ermittelt die richtige for-Anweisung zum angegebenen next. Für Programme, in denen viele for und next Schleifen vorkommen, ist es übersichtlicher, wenn Sie den Namen der Variablen in der next-Anweisung angeben.

ORIGIN

Im vorigen Programm haben wir das Kommando PLOT benutzt, um den Kreismittelpunkt zu zeichnen, und haben dann die x-, y-Koordinaten zu dieser Position addiert. Wir können darauf verzichten, wenn wir das ORIGIN Kommando anwenden. Hiermit bestimmen Sie die Mitte das Kreises, so daß wir nur alle Punkte an der Peripherie des Kreises bestimmen müssen.

Tippen Sie das folgende Programm ein, um dies zu sehen:

```
new
5 cls
10 for a=1 to 360
15 deg
20 origin 320,200
30 plot 190*cos(a),190*sin(a)
40 next
run
```

Sie können Zeile 15 entfernen, oder in Zeile 30 PLOT in DRAW ändern, um einen ausgefüllten Kreis von der Mitte aus zu schreiben.

Das folgende Programm erstellt vier kleinere Kreise, tippen Sie:

```
new
5 cls
10 for a=1 to 360
15 deg
20 origin 196,282
30 plot 50*cos(a),50*sin(a)
40 origin 442,282
50 plot 50*cos(a),50*sin(a)
60 origin 196,116
70 plot 50*cos(a),50*sin(a)
80 origin 442,116
90 plot 50*cos(a),50*sin(a)
100 next
run
```

Sie können weiter ausprobieren, indem Sie Zeile 15 entfernen und in den Zeilen 30, 50, 70 und 90 das DRAW Kommando verwenden.

GOSUB RETURN

Eine Serie von Befehlen, die in einem Programm an verschiedenen Stellen gebraucht werden, kann in einem Unterprogramm (sog. Subroutine) untergebracht werden. Diese Subroutine kann mit dem GOSUB Kommando gefolgt von der Zeilennummer aufgerufen bzw. angesprungen werden.

Sie beenden eine GOSUB-Routine mit dem RETURN Kommando. An dieser Stelle kehrt der Computer zurück zu der Zeile, die auf das zuletzt befolgte GOSUB Kommando folgt.

Im vorigen Programm wurde die Anweisung plot `50*cos(a),50*sin(a)` 4 Mal geschrieben. Diese Anweisung kann als Subroutine geschrieben werden, und mit dem `GOSUB` Kommando aufgerufen werden. Tippen Sie folgendes:

```
new
5 cls
10 for a=1 to 360
15 deg
20 origin 196,282
30 gosub 120
40 origin 442,282
50 gosub 120
60 origin 196,116
70 gosub 120
80 origin 442,116
90 gosub 120
100 next
110 end
120 plot 50*cos(a),50*sin(a)
130 return
run
```

Wir haben das `end` Kommando in Zeile 10 eingefügt, so daß das Programm nicht automatisch nach Anweisung 100 auf Anweisung 120 weiterläuft, das ja nur für die `GOSUB` Aufrufe benötigt wird.

Wir schließen diesen Abschnitt mit dem folgenden Programm, in dem viele Kommandos und Schlüsselwörter vorkommen, die Sie jetzt verstehen sollten. Tippen Sie:

```
new
10 mode 0: border 6: paper 0: ink 0,0
20 gosub 160: for x=1 to 19:locate x,3
30 pen 15: print " "; chr$(238)
40 for t=1 to 50: next t: sound 2,(x+100)
50 next x: gosub 160: for b=3 to 22
60 locate 20,b: pen 7: print chr$(252)
70 cls: gosub 160: next b
80 sound 2,0,100,15,0,0,1
90 gosub 160: border 16,24: locate 20,25
100 pen 14: print chr$(253);
110 for t=1 to 1000: next t
120 border 6: gosub 160: for f=3 to 24
130 locate 10,(25-f): pen 2
140 print chr$(144): cls: gosub 160
150 sound 7,(100-f),5:next f: goto 10
160 locate 10,25: pen 12
170 print chr$(239): return
run
```

SOUND

Klangeffekte werden über einen internen Lautsprecher ausgegeben. Wenn Sie mit einem MP2 Modulator/Stromversorgung und Ihrem Fernseher arbeiten, sollten Sie die Lautstärke des Fernsehgerätes abstellen.

Die Lautstärke im Computer kann mit dem `VOLUME`-Regler auf der rechten Seite des Computers eingestellt werden. Sie können den Ton auch auf den Auxiliary-Eingang Ihrer Stereoanlage geben, indem Sie die (I/O) Buchse auf der Rückseite links an Ihrem Computer benutzen. Sie können dann die Musik in Stereo über Ihre Hifi-Anlage oder über Kopfhörer hören.

Das `SOUND` Kommando hat sieben Parameter; die ersten zwei müssen Sie angeben, die restlichen sind wahlweise. Das Kommando sieht folgendermaßen aus:

`SOUND` Kanalstatus, Tonperiode, Dauer, Lautstärke, Lautstärken-Hüllkurve, Ton-Hüllkurve, Geräuschperiode.

In den folgenden Beispielen setzen wir den Kanalstatus, d. h. seine Referenznummer, auf 1.

TONPERIODE

In Anhang VII sehen Sie, daß die Note mittleres C die Tonperiode 478 hat. Tippen Sie:

```
new
10 sound 1,478
run
```

und Sie hören, wie die Note C 0,2 Sekunden lang ertönt.

DAUER

Wenn Sie die Dauer nicht angeben, wird eine Dauer von 0,2 Sekunden angenommen. Die Einheit für Dauer ist 0,01 Sekunden. Um einen Ton 1 Sekunde lang erklingen zu lassen, müssen Sie 100 angeben; für 2 Sekunden 200, usw.

```
10 sound 1,478,200
run
```

Mittleres C ertönt 2 Sekunden lang.

LAUTSTÄRKE

Hiermit geben Sie die Anfangslautstärke einer Note an. Sie können Werte von 0 bis 7 angeben. Wenn Sie jedoch eine Lautstärkenhüllkurve angeben, ist dieser Bereich 0 bis 15. Falls Sie keinen Wert angeben, wird 4 angenommen.

```
10 sound 1,478,200,3
run
```

Beachten Sie die Lautstärke. Geben Sie jetzt einen größeren Wert an:

```
10 sound 1,478,200,7
run
```

Sie hören jetzt, daß diese Note viel lauter ist.

LAUTSTÄRKEN-HÜLLKURVE

Das Kommando für die Lautstärken-Hüllkurve heißt `env` (envelope volume). Es hat normalerweise 4 Parameter, von denen die letzten drei in bis zu 5 Hüllkurven-Abschnitten vorkommen können. Wir benutzen hier nur einen Abschnitt. Weitere Erklärungen finden Sie in Kapitel 6.

`env` Hüllkurvennummer, Anzahl Schritte, Schrittamplitude (Größe), Schrittzeit.

HÜLLKURVENNUMMER

Diese Nummer wird im SOUND Kommando angegeben, um die Beziehung zu dieser Hüllkurve herzustellen. Der Bereich für die Hüllkurvennummer geht von 0 bis 15.

ANZAHL SCHRITTE

Dieser Parameter wird in Verbindung mit der Schrittzeit benutzt. Sie wollen z. B. 10 Schritte von je 1 Sekunde haben. In diesem Fall ist die Anzahl Schritte 10. Erlaubt sind 0 bis 127 Schritte.

SCHRITTAMPLITUDE

Jeder Schritt kann eine Amplitude von 0 bis 15, bezogen auf den letzten Schritt, haben. Diese 15 Lautstärken sind die gleichen wie im SOUND Kommando. Der Schritt kann jedoch von −128 bis +127 variieren, so daß die Amplitude nicht nur ganz normal rauf und runter zu verändern ist, sondern auch höher als 15, wodurch wir einige außergewöhnliche Effekte erzielen. Die Schrittamplitude kann Werte von −128 bis +127 annehmen.

SCHRITTZEIT

Diese Zahl legt die Zeit zwischen den Schritten fest und wird in Einheiten von 0,01 (einhunderstel) Sekunden gemessen. Der Wertebereich umfaßt 0 bis 255, so daß die längste Zeit zwischen den Schritten 2,55 Sekunden beträgt.

Um mit der Lautstärken-Hüllkurve zu üben, tippen Sie folgendes Programm ein:

```
5 env 1,10,1,100
10 sound 1,284,1000,1,1
run
```

Zeile 10 definiert eine Note mit einer Periode von 284 (Kammerton a), die 10 Sekunden dauert, eine Anfangslautstärke von 1 hat, und die Lautstärkenhüllkurve 1 benutzt. Zeile 5 definiert die Hüllkurve 1 mit 10 Schritten, wobei die Lautstärke jeweils um 1 nach je 1 Sekunde erhöht wird (100 x 0,01 Sekunden).

Ändern Sie Zeile 5 jeweils wie folgt, und hören Sie die unterschiedlichen Effekte der verschiedenen Hüllkurven:

```
5 env 1,100,1,10
5 env 1,100,2,10
5 env 1,100,4,10
5 env 1,50,20,20
5 env 1,50,2,20
5 env 1,50,15,30
```

Probieren Sie schließlich dies aus:

```
5 env 1,50,2,10
```

Sie merken hierbei, daß die Lautstärke nach der Hälfte der Zeit konstant bleibt. Das ist passiert, weil wir 50 Schritte hatten, die jeweils 0,1 Sekunden lang waren. Die gesamte Zeit der Lautstärkenveränderung betrug demnach 5 Sekunden, die Zeit für die Note im `SOUND` Kommando jedoch 10 Sekunden (Zahl 1000).

Probieren Sie jetzt alleine, welche Klangarten Sie erzeugen können.

TON-HÜLLKURVE

Das Kommando für die Ton-Hüllkurve heißt `ent` (envelope tone). Das Kommando `ent` hat normalerweise 4 Parameter. Die letzten 3 können in bis zu 5 möglichen Hüllkurven-Abschnitten vorkommen. Wir benutzen hier nur einen Abschnitt. Weitere Erklärungen finden Sie in Kapitel 6.

`ent` Hüllkurvennummer, Anzahl Schritte, Tonperiode des Schrittes, Schrittzeit.

HÜLLKURVENNUMMER

Diese Nummer wird im `SOUND` Kommando angegeben, um die Beziehung zu dieser Hüllkurve herzustellen. Der Bereich für die Hüllkurvennummer geht von 0 bis 15.

ANZAHL SCHRITTE

Dieser Parameter wird in Verbindung mit der Schrittzeit benutzt. Sie wollen z. B. 10 Schritte von je 1 Sekunde haben. Erlaubt sind 0 bis 239 Schritte.

TONPERIODE DES SCHRITTES

Die Tonperiode darf je Schritt von −128 bis + 127 variieren. Negative Schritte erhöhen die Frequenz der Töne (ergibt höhere Töne). Die kürzeste Tonperiode ist 0. Daran müssen Sie denken, wenn Sie die Ton-Hüllkurve berechnen. Anhang VII enthält eine Übersicht aller möglichen Tonperioden. Die Tonperiode der Schrittzahlen kann Werte zwischen −128 bis +127 annehmen.

SCHRITTZEIT

Diese Zahl legt die Zeit zwischen Schritten fest und wird in Einheiten von 0,01 (einhundertstel) Sekunden gemessen. Der Wertebereich umfaßt 0 bis 255, so daß die längste Zeit zwischen den Schritten 2,55 Sekunden beträgt.

Um mit der Tonhüllkurve zu üben, tippen Sie das folgende Programm ein:

```
5 ent 1,100,2,2
10 sound 1,284,200,15,0,1
run
```

Zeile 10 definiert eine Note mit einer Periode von 284 (Kammerton a), die 2 Sekunden dauert, eine Anfangslautstärke von 15 (maximaler Wert) hat, und die Tonhüllkurve Nummer 1 benutzt.

Zeile 5 definiert Tonhüllkurve 1 mit 100 Schritten, wobei die Tonperiode um 2 alle 0,02 Sekunden (2 hundertstel) erhöht wird (die Frequenz wird dabei reduziert, tiefere Töne).

Ändern Sie Zeile 5 jeweils wie folgt, dann lassen Sie das Programm jedesmal ablaufen (run). Hören Sie sich den jeweiligen Effekt bei der Änderung der Tonhüllkurve an:

```
5 ent 1,100,-2,2
5 ent 1,10,4,20
5 ent 1,10,-4,20
```

Ersetzen Sie jetzt das SOUND Kommando und die Tonhüllkurve durch:

```
5 ent 1,2,17,70
10 sound 1,142,140,15,0,1
15 goto 5
run
```

Drücken Sie zweimal auf [ESC] um die Schleife zu beenden.

Sie können nun die Lautstärkenhüllkurve, die Tonhüllkurve und das **SOUND** Kommando kombinieren, um verschiedene Klänge zu erzeugen. Fangen Sie mit dem folgenden Beispiel an:

```
new
5 env 1,100,1,3
10 ent 1,100,5,3
20 sound 1,284,300,1,1,1
run
```

Dann, ersetzen Sie Zeile 10 durch:

```
10 ent 1,100,-2,3
```

Ersetzen Sie jetzt alle Zeilen wie folgt:

```
5 env 1,100,2,2
10 ent 1,100,-2,2
20 sound 1,284,200,1,1,1
run
```

Probieren Sie jetzt Ihre eigenen Variationen, viel Spaß.

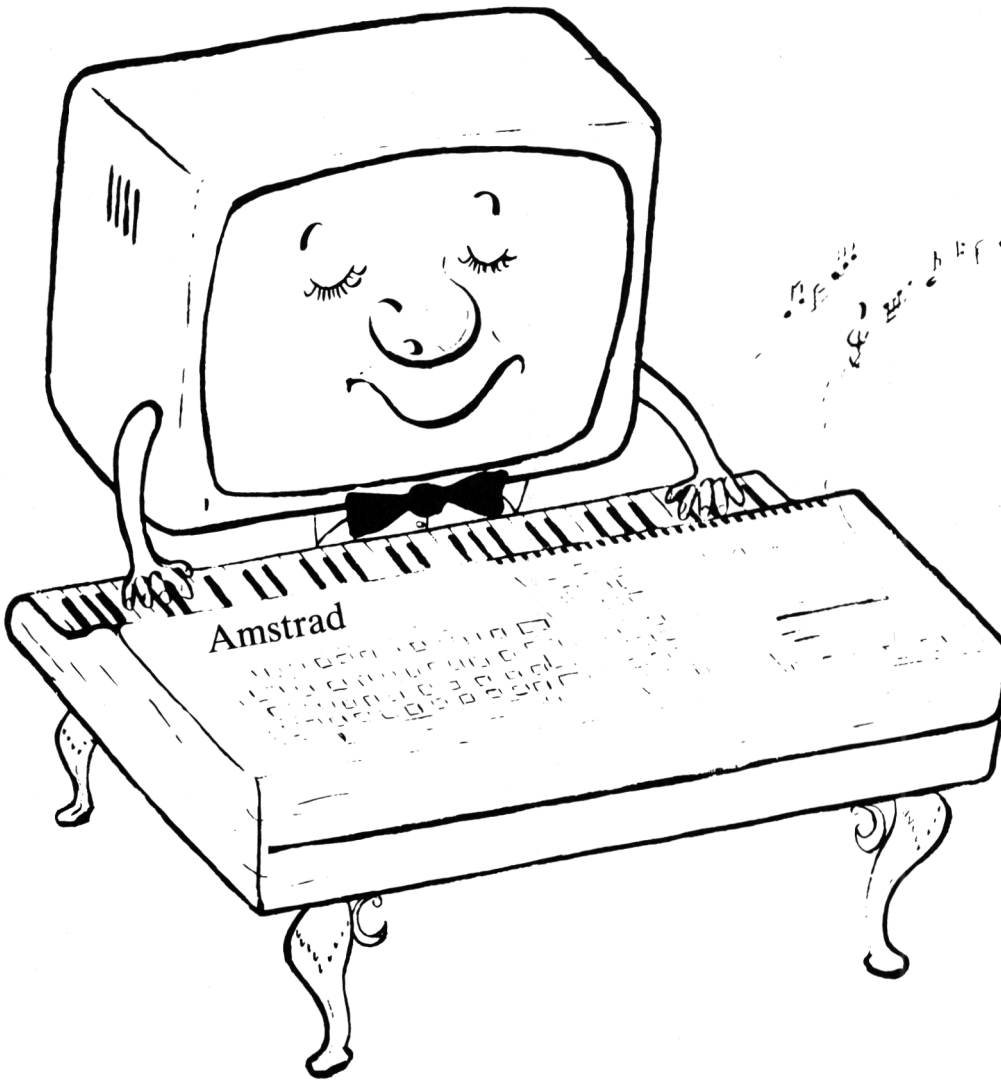
GERÄUSCHE (noise)

Sie können am Ende des SOUND Kommandos auch Geräusche anfügen. Es gibt eine Geräuschbreite im Bereich von 1 bis 31. Probieren Sie es aus, indem Sie eine Geräuschzahl am Ende des **SOUND** Kommandos anfügen, und behalten Sie noch das **ENV** Kommando (Lautstärkenhüllkurve) bei.

Ersetzen Sie die Zeilen 5 und 20, indem Sie tippen:

```
5 env 1,100,3,1
20 sound 1,200,100,1,1,1,5
run
```

Versuchen Sie jetzt noch andere Geräusche zu produzieren. Ändern Sie die Lautstärkenhüllkurve **(ENV)** und das **SOUND** Kommando, mit und ohne Geräusch.



1. So fangen Sie an:

Wenn Sie die Einführung für Anfänger übersprungen haben, in der der Aufbau, das Einschalten und der Umgang mit der Tastatur erklärt wurden, und Sie mit den hier verwendeten Begriffen Schwierigkeiten haben, gehen Sie bitte zurück zur Einführung des Anfängerkurses.

Themen dieses Kapitels:

* Sprachregeln für dieses Buch
* Anschalten
* Vertrautwerden mit der Tastatur

Auch wenn Sie schon Erfahrung mit Computern haben, beachten Sie trotzdem folgende Regeln. Wahrscheinlich haben Sie gerade das Paket geöffnet und können es nicht mehr erwarten, mit dem Computer zu arbeiten. In diesem Kapitel erfahren Sie alles was Sie wissen müssen, um mit BASIC anfangen zu können. Der Neuling sollte jedoch mit dem Kapitel Grundlagen beginnen.

WICHTIG – das MÜSSEN Sie lesen:

Sprachregeln für dieses Handbuch.

Um unterscheiden zu können, ob Tasten auf der Tastatur oder Text in Programmen gemeint ist, gelten ab hier folgende Sprachregeln:

`[ENTER]`: Tasten, bei deren Betätigung kein Zeichen am Bildschirm angezeigt wird, stehen in `[ ]`.

`AXBC`: Tasten, die bei Betätigung ein Zeichen am Bildschirm anzeigen, sind normal geschrieben.

`10 FOR N = 1 to 1000`: Text, der entweder am Bildschirm angezeigt ist oder über die Tastatur eingegeben werden soll, erscheint in Schreibmaschinenschrift. Beachten Sie den Unterschied zwischen der Null `0` und dem großen Buchstaben O.

Es wird vorausgesetzt, daß Sie jedes Programm oder jeden direkten Befehl durch Drükken der Taste `[ENTER]` abschließen. Diese Angabe wird im Rest des Buches nicht mehr angezeigt.

Desweiteren wird vorausgesetzt, daß Sie `run` eintippen, wenn ein Programm ausgeführt werden soll. BASIC wandelt alle Schlüsselwörter, die in Kleinbuchstaben eingegeben wurden, in Großbuchstaben um, wenn das Programm mit `LIST` angezeigt wird. Die Beispiele von hier an werden mit Großbuchstaben geschrieben. wenn Sie aber Schlüsselwörter in Kleinbuchstaben eingeben, können Sie Fehler in solchen Wörtern leicht erkennen, da fehlerhafte Schlüsselwörter mit `LIST` nicht umgewandelt werden.

1.1.1 Öffnen des Kartons

Der Farbbildschirm

WICHTIG

Lesen Sie bitte die genaue **Aufbauanleitung** am Anfang dieses Handbuches.

Wenn der Computer, wie in Abbildung 1 gezeigt, angeschlossen ist, schalten Sie zuerst den Bildschirm und dann den Computer mit dem Schiebeschalter auf der rechten Seite ein. Nach etwa 30 Sekunden erscheint folgendes Bild:

```
Amstrad 64K Microcomputer (v1)

© 1984 Amstrad Consumer Electronics plc
          and Locomotive Software Ltd.

BASIC 1.0

Ready
```

Dieses Bild wird immer gezeigt, wenn der Computer angeschaltet wird oder wenn man folgende drei Tasten in der angegebenen Reihenfolge drückt und festhält: [CTRL], [SHIFT], [ESC]. Dadurch befindet sich der Computer in seinem Ursprungszustand. Üben Sie jetzt den Kaltstart mit den 3 Tasten!

Stellen Sie nun die Helligkeit mit BRIGHTNESS, rechts unten am Bildschirm, ein. Die Farbe ist bereits vom Werk vorgegeben. Wenn Sie diese ändern wollen, (gelbe Schrift auf blauem Grund) müssen Sie dies per Programm tun. Siehe Graphik Anweisungen in Kapitel 8.

Wenn es Ihnen nichts ausmacht, dem Lehrstoff ein wenig vorauszueilen, finden Sie hier ein sehr einfaches kurzes Programm, das die farbige Texteingabe in dem hochauflösenden 80 Zeichen/Zeile Modus darstellt.

Geben Sie ein:

```
10 MODE 2
20 INK 1,0
30 INK 0,13
40 BORDER 13
```

...Sie können dieses Programm auch als Einzelanweisung im 'Direkt-Modus' eingeben.

Wenn Sie dieses Programm nicht verstehen oder nicht eintippen können, müssen Sie zurück zur Einführung für Anfänger gehen. Die 80 Zeichen/Zeile Anzeige ist die komfortabelste zur Programmentwicklung. Sie können das Programm, wenn Sie Kapitel 2 gelesen haben, am Anfang einer Leercassette speichern, um sich die Eingabe bei Wiederholungen zu sparen.

VORSICHT! Vermeiden Sie zu nahe am Bildschirm zu sitzen, oder die Helligkeit zu hoch einzustellen. Es könnte Ihren Augen wehtun. Fühlen Sie Augenschmerzen, schalten Sie ab und tun Sie etwas anderes.

1. Schalten Sie immer eine genügend helle Lichtquelle zu, damit Sie dieses Handbuch leicht lesen können.
2. Stellen Sie die geringstmögliche Helligkeit ein.
3. Sitzen Sie weit genug vom Bildschirm entfernt.

1.1.2 Der Grünmonitor

Der GT65 Bildschirm hat 3 Regler unterhalb der Anzeige. `Helligkeit`, `Kontrast` und vertikale `Verschiebung` können damit eingestellt werden. Die vertikale `Verschiebung` braucht normalerweise nur einmal eingestellt zu werden. `Helligkeit` und `Kontrast` hängen ab von den jeweiligen Lichtbedingungen unter denen der CPC464 betrieben wird.

Benutzt man einen einfarbigen Bildschirm, erscheint die gleiche Meldung nach dem Start wie bei dem Farbbildschirm, allerdings in hellem Grün auf dunklem Grund. Verschiedene Farben werden bei einem einfarbigen Bildschirm durch Schattierungen der Grundfarbe dargestellt.

Obwohl zuviel Arbeit am Bildschirm Augenschmerzen verursachen kann, ist ein einfarbiger Bildschirm augenschonender als der Farbbildschirm, da die Auflösung (d. h. Anzahl von Bildpunkten) normalerweise beim einfarbigen Bildschirm besser ist. Das Bild ist stabil und flackert nicht. Regeln Sie `Helligkeit` und `Kontrast` entsprechend den Umweltbedingungen.

Um den Bildschirm in den 80 Zeichen/Zeile Modus zu bringen, tippen Sie folgendes Programm ein und speichern Sie es auf Cassette ab:

```
10 REM setze Anzeigeformat
20 FOR n=0 TO 26
30 MODE 2
40 INK 1,n
50 INK 0,(26-n)
55 BORDER n
60 LOCATE 15,12: PRINT "druecke irgendeine Taste
   um das Anzeigeformat zu aendern"
70 a$=INKEY$
80 IF a$="" GOTO 70
90 NEXT
100 GOTO 20
```

Obiges Programm zeigt eine weitere Besonderheit, die dieses Handbuch betrifft.

Manchmal werden bei `LIST` von Programmen Teile einer Zeile in eine zweite übertragen. Leerzeichen dienen dabei lediglich der besseren Lesbarkeit und belegen im Programm keinen Speicherplatz. Damit sind noch nicht alle Möglichkeiten der Darstellung ausgeschöpft, aber Sie haben einen Eindruck davon bekommen. Wenn Ihnen eine Kombination gut gefällt, drücken Sie die Taste `[ESC]` zweimal und das Programm bleibt stehen. `*Break*` wird angezeigt.

Im Folgenden werden viele Anmerkungen vorkommen, die sich lediglich auf Farbbildschirmdarstellung beziehen. Programme mit interessanten Farb- und Graphikeffekten können auf einem einfarbigen Bildschirm nahezu unsichtbar sein, obwohl große Mühe darauf verwendet wurde, mit Grau- oder Grünschattierungen gleiche Effekte zu erzielen (siehe Kapitel 5).

Der Vorteil des grünen Bildschirms ist die klare, scharfe Darstellung, speziell bei der Programmentwicklung. Aber wenn Sie mit dem Farbbildschirm einmal gearbeitet haben, wollen Sie wahrscheinlich dabei bleiben!

1.1.3 Der MP2 Modulator/Stromversorgung

Der MP1 Modulator/Stromversorgung ist ein Zusatz, den Sie kaufen können, um anstelle des grünen GT65 Ihren normalen Farbfernseher verwenden zu können.

> WICHTIG
>
> Lesen Sie bitte die genaue **Aufbauanleitung** am Anfang dieses Handbuches.

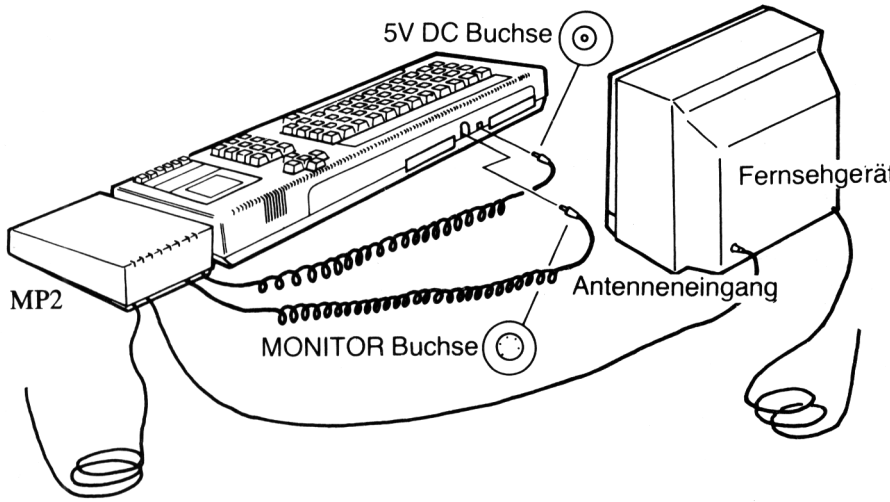


Bild 2: Anschluß von MP2, Computer und Farbfernseher.

Der MP2 Zusatz sollte rechts der Tastatur plaziert werden wie in Bild 2 gezeigt.

Reduzieren Sie jetzt die Lautstärke am Fernseher soweit wie möglich (der CPC464 hat seinen eigenen Lautsprecher). Das Geräusch, das Sie dann hören, ist ganz normal. Schalten Sie erst das Fernsehgerät ein, und dann schalten Sie den Computer mit dem Schiebeschalter an der rechten Seite (POWER) ein.

Die rote Lampe (In Function) rechts oben auf der Tastatur sollte aufleuchten, und Sie müssen jetzt den Fernseher einstellen, um das Signal vom Computer zu empfangen.

Wenn Sie ein Fernsehgerät haben, bei dem Sie die Kanäle über Drucktasten wählen können, so drücken Sie die Taste eines freien oder nicht benutzten Kanales. Stellen Sie die Senderwahl auf ungefähr Kanal 36, bis Sie ein Bild bekommen, das so aussieht wie das auf der folgenden Seite:

Justieren Sie den Fernseher so, daß die größte Bildschärfe erreicht wird. Die Schrift ist goldgelb auf tiefblauem Hintergrund, kann aber auch je nach Einstellung ihres Fernsehgerätes etwas abweichen.

Wenn Ihr Fernsehgerät einen Drehwähler für die Programmauswahl hat, so drehen Sie solange bis obiges Bild klar und ruhig erscheint (auch hier ungefähr bei Kanal 36).

Da das Signal öfters umgeformt wird, erst moduliert, dann demoduliert, kann die Qualität des Video-Signals darunter leiden. Das Bild kann nicht so gut sein wie bei einem Video-Monitor, und abhängig von der Qualität Ihres Fernsehgerätes kann es sein, daß der 80-Spalten-Modus (Modus 2) keine zufriedenstellenden Ergebnisse liefert. In diesem Fall sollten Sie möglichst Modus 1 benutzen.

1.2 Die ersten Schritte

Die Geräte sind angeschlossen, das Netzkabel eingesteckt, der Monitor ist angeschlossen, der Computer ist eingeschaltet und wartet auf Ihre Eingabe.

Die „Einschalt"-Meldung ist der einzige „eingebaute" Text, der erscheinen kann, ohne daß Sie irgendwelche Tasten drücken müssen. Wenn Sie BASIC bereits kennen, haben Sie vielleicht schon ein kleines Programm eingegeben, um mit dem System „warm zu werden". Amstrad BASIC wird Ihnen in vielerlei Hinsicht vertraut vorkommen.

1.2.1

Um Sie ein wenig einzuführen, zeigen wir Ihnen ein kleines Programm, das die eingebauten Zeichen des Computers anzeigt. Das ist der Zeichenvorrat, der über die Tastatur eingegeben und über Bildschirm und Drucker ausgegeben werden kann.

Einige der Zeichen, die Sie sehen, können nicht direkt über die Tastatur eingegeben werden, sondern nur mit Hilfe der `PRINT CHR$` (Zahl) Funktion gedruckt werden, die später beschrieben wird.

Das kommt daher, daß alle Informationen in einem Computer in sogenannten Bytes gespeichert werden. Im Anhang II können Sie sehen, daß ein Byte 256 verschiedene Kombinationen von Zeichen darstellen kann. Es können also 256 unterschiedliche Zeichen gebildet werden, von denen aber auf den üblichen Tastaturen und Druckern nur ca. 96 verwendet werden.

Der Standardbereich des Zeichenvorrates ist eine Untermenge. Er wird überall in der Computerwelt als das 'ASCII' System bezeichnet, das sich ableitet aus:

American
Standard
Code for
Information
Interchange

Damit wird erreicht, daß Daten von einem Computer zu einem anderen übertragen werden können. Das ist so ziemlich das Einzige im Computerwesen was eindeutig definiert ist. Wir empfehlen Ihnen daher, sich mit dem ASCII-Zeichensatz vertraut zu machen. Anhang III zeigt den ASCII-Zeichensatz und alle weiteren Zeichen, die es auf dem CPC464 gibt, mit ihrem numerischen Wert.

Einige der 'nicht druckbaren' Zeichen können angezeigt werden, wenn Sie die Kontroll-(`[CTRL]`)Taste zusammen mit einer anderen Taste drücken. Beschäftigen Sie sich damit aber jetzt nicht, bis Sie die Bedeutung dieser Tastenkombinationen kennen.

1.2.2

Damit Sie genau sehen können, wie diese Zeichen aussehen, tippen Sie das folgende Programm ein und wir werden Ihre Neugier und den CPC464 herausfordern. Dabei sehen Sie auch wie einfach Programme erstellt werden können und daß es keine Probleme mit der Hardware gibt, da Sie ja durch Ein- und Ausschalten immer wieder die Start-Meldung bekommen können. Der Computer CPC464 wartet einfach darauf, von Ihnen programmiert und benutzt zu werden.

(Wenn Sie beim Eingeben eines Programmes einen Fehler machen, gehen Sie zu 1.2.7. Dort sehen Sie, wie man Korrekturen durchführt ohne neu zu starten und das Programm neu eingeben zu müssen).

Groß- und Kleinbuchstaben spielen bei der Programmeingabe keine Rolle. Die Worte müssen lediglich durch `Leerstellen, Kommas, Strichpunkte` oder andere gültige Trennzeichen voneinander getrennt werden, da Amstrad BASIC die Verwendung von reservierten Wörtern in Variablennamen erlaubt (siehe Anhang VIII).

Der Tastatur-„Körper“ ist in Bild 4 dargestellt. Dabei handelt es sich um die Originaltastatur. Viele der Tasten können mit anderen Werten versehen werden, wie in den nachfolgenden Kapiteln gezeigt wird.

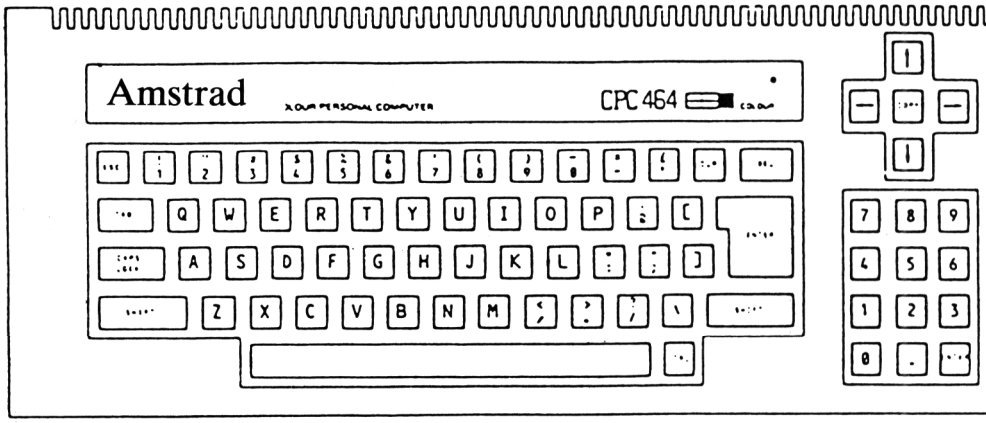


Bild 4: Die Tastatur des CPC464

Wenn Sie [ENTER] drücken, wird die Programmzeile oder eine Kommandozeile abgeschlossen und eingegeben. Ohne Zeilennummer wird diese Eingabe sofort verarbeitet, mit Zeilennummer wird die Eingabe als Programmzeile gespeichert.

[ENTER] wird auch manchmal als Carriage Return oder einfach als Return bezeichnet. Diese Bezeichnung stammt noch aus der Zeit, als Computer-Terminals wie mechanische Schreibmaschinen aufgebaut waren. Der Ausdruck ist geblieben, und so wird im ASCII-Zeichensatz der CODE für [ENTER] mit den Buchstaben ‘CR’ bezeichnet. Geben Sie ein:

```
10 FOR N = 32 TO 255
20 PRINT CHR$(N);
30 NEXT N
RUN
```

1.2.3

Sehen Sie was am Bildschirm geschehen ist:

```
Amstrad  64K Microcomputer  (v1)
©1984

 BASIC 1.0

Ready
10 FOR N = 32 to 255
20 PRINT CHR$(N);
30 NEXT N
run
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFG
HIJKLMNOPQRSTUVWXYZ[\]↑ `abcdefghijklmno
pqrstuvwxyz{|}~▒ ▘▝▀▖▌▞▛▗▚▜▄▙▟▐▜▟▛▙▄▟▜▛▙▄▟▜▛▙▄▟▜
```

Einfach indem Sie N in dieser Schreibweise verwenden, weiß BASIC, daß N der Name einer Variablen ist. Die Anweisung FOR erwartet nach dem Schlüsselwort eine Variable und akzeptiert damit die nachfolgende Buchstabenkombination als Variable, wenn sie den Regeln entspricht.

Der Ausdruck N = 32 to 255 bewirkt, daß der Bereich für die Variable N von 32 bis 255 geht. Nachdem die Variable definiert ist, muß dem Computer gesagt werden, was mit ihr geschehen soll, wie z. B. in der nächsten Zeile:

```
20 PRINT CHR$(N);
```

Diese Anweisung bewirkt, daß die Funktion CHR$ das dem augenblicklichen Wert von N entsprechende Zeichen druckt.

Der Strichpunkt am Zeilenende verhindert, daß nach jedem Zeichen, das angezeigt wird, eine neue Zeile angefangen wird. Damit wird eine Zeile Zeichen für Zeichen gefüllt und angezeigt.

Die nächste Anweisung sagt dem Computer, daß er nach Erledigung der Aufgabe mit dem ersten Wert aus der Reihe (32) für die Variable N zu der Anweisung FOR zurückgehen soll. Dann soll die Anweisung mit dem nächst (NEXT) höheren Wert der Variablen N durchgeführt werden.

Diesen Vorgang nennt man Schleife (Loop), eine sehr wichtige Technik beim Programmieren. Schleifen gibt es in jeder Programmiersprache. Sie vereinfachen viele Vorgänge. Wenn Sie die Vorzüge erkannt haben, werden Sie sie sicherlich häufig einsetzen.

Wenn diese FOR-Schleife ihren Grenzwert (255) erreicht hat, wird mit der Anweisung weitergefahren, die auf NEXT folgt. Nachdem hier keine weitere Anweisung mehr kommt, hört das Programm einfach auf und BASIC meldet sich mit READY.

Jetzt können Sie weitere Anweisungen eingeben oder das gleiche Programm nochmal laufen lassen (RUN). Das Programm bleibt solange im Speicher, bis Sie dem Computer audrücklich etwas anderes sagen oder ihn ausschalten. Sollten Sie das Programm nicht auf Cassette gespeichert haben, ist es dann für immer verloren.

Dieses Programmbeispiel zeigt eine elementare Eigenschaft des Computers:

Alles was der Computer intern tut, beruht auf Zahlen. Der Computer hat seinen Zeichenvorrat ausgedruckt, indem er jeweils eine Zahl für das gewünschte Zeichen benutzt hat. Wenn Sie z. B. die Taste A drücken, erkennt der Computer anhand des entsprechenden Wertes der Taste in seinen Tabellen, daß diese Taste ein A darstellt und gibt ein entsprechendes Zeichen aus. Jedes Zeichen hat seine interne Nummer, die Sie in Anhang III nachschlagen können.

1.2.4

Verzweifeln Sie nicht, wenn Sie nicht alle Ausdrücke und technischen Angaben auf dieser Seite verstehen. Technisch Interessierten wird dieser Absatz helfen zu verstehen, was der Computer mit Ihren Anweisungen macht und wie er zum gewünschten Ergebnis kommt. Wenn dieses Kapitel für Sie zu schwer ist, gehen Sie einfach weiter zu Kapitel 1.2.5

Der Wert für den Buchstaben A z. B. ist 97. Der Computer versteht auch 97 nicht, sondern setzt diese Zahl in den sogenannten Maschinencode um. Anhang II erklärt den Maschinencode.

Die Umsetzung von dezimalen Zahlen aus unserem täglichen Gebrauch in den Maschinencode (hexadezimale Zahl) erscheint zunächst schwer verständlich, da wir an unser Zahlensystem gewöhnt sind.

Die gleiche Selbstverständlichkeit sollten Sie beim Umgang mit hexadezimalen Zahlen erreichen, dann fällt Ihnen vieles leichter und wird verständlicher.

Der Computer übersetzt ein eingetipptes `A` in eine Zahlendarstellung, die er versteht, und entnimmt dann einer Tabelle eine andere Folge von Ziffern, die das Zeichen definiert. Das Zeichen, das man am Bildschirm sieht, wird durch einen Datenblock erzeugt, der im Speicher als numerische Matrix abgelegt ist.

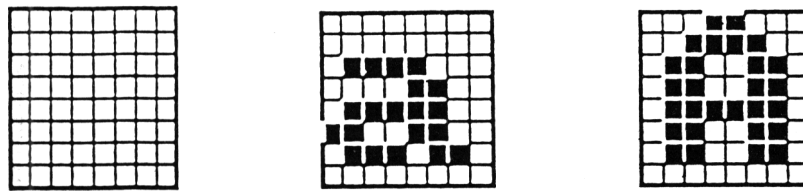


Bild 5: Raster für ein Leerzeichen, kleines a, großes A

Die Elemente dieser Matrix sind Spalten und Zeilen mit Punkten. Das Zeichen wird durch ein entsprechendes Punktmuster dargestellt, das im Computer gespeichert ist. 8 Zeilen und 8 Spalten bilden beim CPC464 ein Zeichen (dargestellt durch je ein Byte). Wenn Sie ein bestimmtes Zeichen nicht finden, können Sie das durch die Anweisung `SYMBOL` oder `SYMBOL AFTER` (beschrieben in Kapitel 8) selbst definieren.

Selbst definierte Zeichen können aus Kombinationen von 0 bis 64 Punkten dargestellt werden.
Überlegen Sie mal wieviele Möglichkeiten das sind!
Sie können Blöcke von Zeichen zusammenfassen, um so ein großes Zeichen zu bilden. Sie sehen, die Anzahl der Möglichkeiten ist begrenzt allein durch Ihre Zeit und Ihren Einfallsreichtum.

1.2.5 Zurück zum Programm!

Das Ergebnis des ersten Programms, das Sie eingegeben haben, ist ziemlich unordentlich. Die Kaltstart-Meldung ist immer noch angezeigt. Es sieht besser aus, wenn Sie den Bildschirm löschen, bevor Sie etwas anzeigen. Mit einer zusätzlichen Anweisung kann dies geschehen.

Tippen Sie folgendes in die Zeile ein, in der der Cursor steht. Wenn Sie nicht wissen, was der Cursor ist (nämlich der Lichtpunkt, der die Stelle anzeigt an der das nächste Zeichen geschrieben wird), sollten Sie lieber doch zuerst die Einführung lesen!

```
5 CLS
RUN
```

Jetzt wird der ganze Bildschirm gelöscht, bevor die neue Anzeige erscheint.

Das zeigt ein weiteres Merkmal von BASIC. Die Reihefolge der Anweisungen ist unerheblich. Das Programm wird in der Reihenfolge der Zeilennummern ausgeführt, auch wenn die Programmliste nicht am Bildschirm sichtbar ist. Neue Anweisungen werden an die richtige Stelle gebracht und im Speicher aufbewahrt.

1.2.6 LIST

Sie können jederzeit mit Hilfe der Anweisung LIST das Programm im Speicher auf dem Bildschirm anzeigen. Geben Sie ein:

```
LIST
```

das Ergebnis ist dann:

```
5 CLS
10 FOR N = 32 TO 255
20 PRINT CHR$(N);
30 NEXT N
Ready
```

Das Programm bleibt im Speicher des CPC464 bis Sie entweder

* ausschalten
* die Tasten [CTRL] [SHIFT] [ESC] nacheinander gedrückt halten
* ein Programm von Cassette laden (LOAD oder RUN)
* den Befehl NEW [ENTER] eingeben, wodurch der Programmspeicher zurückgesetzt wird (nicht der Modus und nicht die Farbe)

Jetzt legen Sie eine Funktionstaste fest, die folgende Aufgabe bekommen soll: [ENTER]CLS:LIST[ENTER]. Damit kann man die Programmentwicklung enorm beschleunigen. Geben Sie ein:

```
KEY138,CHR$(13)+"CLS:LIST"+CHR$(13)
```

Drücken Sie jetzt den Punkt im Ziffernblock der Tastatur. Bis zu 32 Tasten können auf diese Weise vorprogrammiert werden. Weitere Einzelheiten unter Anweisung KEY im Kapitel 8.

```
KEY138,CHR$(13)+"CLS:LIST"
```

dann können Sie den Nummernbereich, den Sie LISTen wollen, angeben. (Zweimaliges Drücken der Taste zeigt die volle Liste).

Wenn Sie viel mit Farben experimentieren, kann es passieren, daß Schrift und Hintergrund die gleiche Farbe haben und nichts mehr lesbar ist. Haben Sie aber folgende Funktion definiert:

```
KEY139,CHR$(13)+"mode2:ink1,0:ink0,9"+chr$(13)
```

so müssen Sie nur die kleinere der beiden [ENTER] Tasten (diejenige im Ziffernblock) drücken, und Sie kommen zu zwei verschiedenen Farben zurück, ohne daß Sie Ihr Programm verlieren.

Diese Tastendefinitionen werden beim Zurücksetzen der Maschine gelöscht. Wenn Sie also Ihre Tastenfunktionen definiert haben, so speichern Sie diese als Programm auf eine Cassette, damit Sie sie immer wieder verwenden können.

1.2.7 Programmbearbeitung

Sie werden bestimmt einmal Fehler beim Eingeben machen. Mit dem CPC464 ist es sehr einfach, Tippfehler zu korrigieren, ohne dabei versehentlich solche Zeilen zu überschreiben, die man gar nicht ändern will.

Die Pfeiltasten, die den Cursor steuern, helfen Ihnen dorthin zu gelangen, wo Änderungen nötig sind.

Sie wollen z. B. in folgender Zeile

```
10 FOR N = 332 TO 255
```

etwas ändern. Dann haben Sie mehrere Möglichkeiten:

1. Sie drücken [ENTER] und geben die ganze Zeile neu ein. Die alte Zeile wird durch die neue mit der gleichen Zeilennummer im Speicher überschrieben.

2. Mit der Taste [←] bringen Sie den Cursor an die zu ändernde Stelle.

```
10 FOR N = 332 TO 255
```

Das Zeichen, auf dem sich der Cursor befindet, ist invertiert dargestellt. Das heißt, die Farbe der Schrift wurde gegen die Untergrundfarbe ausgetauscht.

Wenn Sie jetzt [CLR] (Abkürzung für *CLEAR*) drücken, wird das Zeichen gelöscht und der Rest der Zeile wird nach vorne gerückt.

```
10 FOR N = 32 TO 255
```

Drücken Sie jetzt [ENTER], und diese korrigierte Zeile ist die neue Programmzeile. Es spielt keine Rolle, wo sich der Cursor dabei befindet. Es wird generell die ganze Zeile übernommen.

3. Sie können den Cursor unmittelbar hinter das zu löschende Zeichen plazieren:

```
10 FOR N = 332 TO 255
```

Wenn Sie jetzt [DEL] (Abkürzung für *DELETE*) drücken, wird das Zeichen links vom Cursor gelöscht und die Zeile wieder nach links ausgerichtet.

Drücken Sie [ENTER] und die Zeile wird so wie vorher gespeichert.

```
10 FOR N = 32 TO 255
```

1.2.8 Nachträgliche Korrekturen

Obige Methoden sind dann geeignet, wenn Sie den Fehler bemerken, bevor Sie mit [ENTER] die Zeile abgeschlossen haben. Dies ist aber nicht immer der Fall. Viele Fehler bemerken Sie erst, wenn das Programm abläuft und der Computer mit einer Fehlermeldung (siehe Anhang VIII) antwortet.

Bei einer Reihe von Fehlern wird die fehlerhafte Zeile angezeigt, wobei der Cursor auf dem 1. Zeichen (ganz links) steht. Sie können dann mit den oben gezeigten Methoden den Fehler so beheben, als hätten Sie ihn vor dem Abschicken der Programmzeile entdeckt. Wird die fehlerhafte Zeile nicht ausgegeben, so müssen Sie mit LIST das Programm anzeigen, den Fehler suchen und mit den beschriebenen Methoden beheben.

1.2.9 Der COPY CURSOR

Zeigen Sie zuerst mit dem LIST Befehl das Programm an. Wir gehen davon aus, daß Sie noch mit dem kleinen Beispielprogramm arbeiten.

```
5 CLS
10 FOR N = 32 TO 255
20 PRINT CHRS(N);
30 NEXT N
```

Der Fehler ist in der Zeile 20, in der ein S anstelle des $-Zeichens steht ($ sagt dem Computer, daß die unmittelbar folgenden Zeichen als Text und nicht als numerische Daten zu verarbeiten sind).

Sie können jetzt Zeile 20 neu eingeben oder den Screen-Editor (Bildschirm-Bearbeitung) wie folgt benützen:

Halten Sie eine der [SHIFT] Tasten gedrückt und drücken Sie dann die entsprechende Cursortaste nach unten [↓] oder nach oben [↑], je nachdem wohin Sie müssen.

Der *'COPY Cursor'* bewegt sich dadurch vom Hauptcursor weg (die beiden haben die gleiche Form) und wird in die Zeile gebracht, die Sie ändern wollen. Positionieren Sie den COPY Cursor auf das 1. Zeichen der Zeile.

```
5 CLS
10 FOR N = 32 TO 255
20 PRINT CHRS(N);
30 NEXT N
Ready
■
```

Wenn Sie ohne Drücken der [SHIFT] Taste den Cursor in die zu ändernde Zeile bringen, nützt das nichts, weil der Computer nur die Zeichen annimmt, die nach dem Hauptcursor eingegeben werden.

Sollten Sie das doch einmal tun und auf diese Weise die Zeile überschreiben, können Sie sich leicht helfen, indem Sie die [ESC] Taste drücken *BEVOR* Sie eine [ENTER] oder Funktionstaste, die ein CHR$(13) enthält, drücken.

Wenn Sie zufällig das Kommandowort NEW eintippen und mit [ENTER] abschicken, so ist Ihr ganzes Programm für immer weg – also Vorsicht!

Wenn Sie eine Zeile verlassen wollen ohne [ENTER] gedrückt zu haben, reagiert der Computer mit einem Piepston. Drücken Sie [ENTER] und das Problem ist behoben. Sollten Sie allerdings eine Zeilennummer angegeben haben, ist diese Zeile überschrieben und muß neu eingegeben werden.

Ausgehend vom ersten Zeichen der Zeile, die zu ändern ist, gehen Sie mit der Taste [COPY] bis zu der Stelle, die Sie ändern wollen. Durch die Wiederholfunktion (Gedrückthalten der Taste) kann der Cursor sehr schnell bewegt werden.

```
5 CLS
10 FOR N = 32 TO 255
20 PRINT CHRS(N);
30 NEXT N
Ready
20 PRINT CHR■
```

Das $-Zeichen wird dort geschrieben, wo sich der Hauptcursor befindet. Dieser bewegt sich dadurch um eine Stelle nach rechts.

```
20 PRINT CHR$■
```

Bringen Sie jetzt den COPY Cursor hinter das fehlerhafte S, indem Sie wieder die [SHIFT] Taste gedrückt halten und einmal auf die Cursor-Rechts-Taste drücken. Der COPY-Cursor bleibt jetzt über der (stehen. Lassen Sie [SHIFT] los und drücken Sie solange die [COPY] Taste, bis Sie über das Ende der Zeile hinausgekommen sind. Drücken Sie jetzt [ENTER], und die korrigierte Zeile im Bildschirm unten ersetzt die fehlerhafte aus dem Listing.

Sie können auch mit [COPY] die ganze Zeile kopieren und dann ganz normal die kopierte Zeile verbessern. [CTRL] und [←] oder [→] bringen den Cursor in einem Schritt an den Zeilenanfang oder das Zeilenende.

Üben Sie, es geht ganz einfach.

Schließlich gibt es noch die Methode der direkten Zeilennummernangabe. Tippen Sie:

```
EDIT 20
```

Der Computer antwortet mit:

```
20 PRINT CHRS(N);
```

Gehen Sie dann nach der Methode vor, wie in 1.2.7 beschrieben. Wenn Sie Schwierigkeiten haben, drücken Sie [ESC], geben dann LIST ein und Sie haben das Programm wieder. Die Zeile, in der [ESC] gedrückt wurde, ist unverändert.

Nach Abschluß der Korrektur tippen Sie LIST und Sie sehen die neue Programmversion. Wenn noch Fehler darin sind, versuchen Sie es noch einmal.

Geben Sie folgendes ein, um auch die anderen zwei Modes kennenzulernen:

```
MODE 0
RUN
```

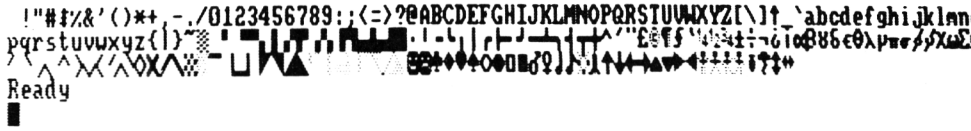
Beachten Sie, daß zuerst der Bildschirm gelöscht wird, und dann Ihr Programm pro Zeile 20 Zeichen ausgibt.

Um zum vorhergehenden Modus zurückzukehren geben Sie ein:

```
MODE 1
RUN
```

Um nun zur 80 Zeichen/Zeile Anzeige zu kommen, tippen Sie ein:

```
MODE 2
RUN
```



Sie haben nun den Anfang gemacht. Ihre erste Neugierde sollte befriedigt sein, dafür aber Ihr Appetit richtig angeregt. Erfahrene Benutzer werden jetzt überlegen, welches Programm sie zuerst umstellen oder schreiben sollen. Anfänger sollten in diesem Handbuch weiterlesen, um sich mit der Maschine und den maschinenspezifischen BASIC Befehlen vertraut zu machen.

Vorstellung des CPC464 Kapitel 1/Seite 17

2 Cassetten-Datacorder

Laden und Bedienen des Recordersystems

Themen dieses Kapitels:

* Gemeinsamkeiten und Unterschiede zwischen Daten- und Toncassetten
* Laden von Daten von der Cassette
  – Das Begrüßungsband
* Die Wahlmöglichkeit bei der Recordergeschwindigkeit
* Speichern von Programmen auf Cassette
* Lesefehler

Der CPC464 kann Daten nur speichern, solange er eingeschaltet ist und Strom zugeführt wird. In der Fachsprache wird das als 'flüchtiger Speicher' bezeichnet. Wenn Sie Daten und Programme länger speichern wollen, müssen Sie dies auf Cassetten oder einem zusätzlich zu erwerbenden Diskettensystem tun.

2.1. Bedienung des Datacorders

Rechts von der Eingabetastatur des Rechners finden Sie den Datacorder – Bild 2.1. Der mechanische Aufbau ist ähnlich wie bei einem Cassettenrecorder, die Elektronik wurde aber speziell auf die Bedürfnisse der Datenspeicherung zugeschnitten.

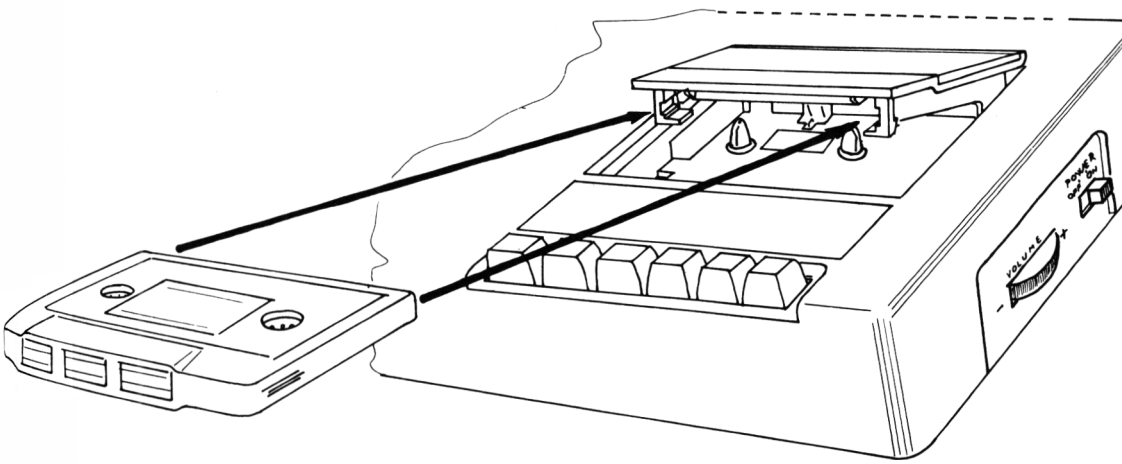


Bild 2.1 So wird die Cassette richtig eingelegt

Die Tastatur und deren Funktion ist dieselbe wie bei normalen Cassettenrekordern:

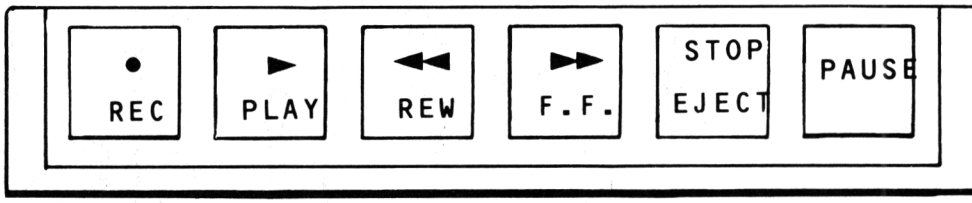


Bild 2.2 Die Tastatur des CPC464 Datacorders

Anmerkung: Diese Tasten müssen fester gedrückt werden als die der Eingabetastatur.

[REC] = Aufnahme von Daten oder Programmen. Zum Aufnehmen müssen die Schreibschutzkerben der Cassette abgedeckt und der Deckel des Cassettenfachs geschlossen sein. Halten Sie die [REC] Taste niedergedrückt und drücken Sie dann auf [PLAY]. Jetzt kann durch einen Programmbefehl oder die direkte Eingabe des Kommandos SAVE auf die Cassette aufgenommen werden.

[PLAY] = Abspielen, hier Laden oder Lesen von Programmen oder Daten von der Cassette entweder durch ein direktes Kommando oder per Programm. Sowohl [REC] als auch [PLAY] werden bei Bandende automatisch abgeschaltet.

[REW] = Rückspulen der Cassette von rechts nach links. Keine automatische Abschaltung am Bandende. Lassen Sie das Gerät in dieser Funktion nicht unbeaufsichtigt, der Motor könnte Schaden leiden.

[F.F.] = Schnelles Vorspulen nach rechts. Wie bei [REW] keine Abschaltung am Bandende.

[STOP/EJECT] = Stoppt alle Cassettenoperationen und bringt alle Tasten in Ausgangsstellung. Bei zweimaligem Drücken öffnet sich der Deckel.

[PAUSE] = Mechanische Pause zusammen mit der [REC] und [PLAY] Funktion. Benutzen Sie diese Taste nie beim Aufnehmen oder Einlesen von Daten, sonst gibt es Fehler! Alle Pausen beim Einlesen oder Aufnehmen werden durch interne Anweisungen der Software gesteuert, die mechanische Pausenfunktion bleibt dabei ungenutzt.

2.2 Schreibschutz (Bild 2.3)

Um unbeabsichtigtes Löschen von Cassetten zu verhindern, haben alle Cassetten einen mechanischen Schreibschutz. Wenn Sie die Abdeckung mit einem Messer entfernen, können Sie, mit dieser Cassette im Laufwerk, die Taste [REC] nicht mehr drücken und deshalb die Cassette auch nicht mehr beschreiben.

Jede Cassettenseite hat ihren eigenen Schreibschutz. Wenn Sie eine schreibgeschützte Cassette wieder neu beschreiben wollen, kleben Sie die offene Kerbe einfach mit Klebeband zu.

2.3 Laden der Cassette

Bild 2.1 zeigt, wie eine Cassette richtig in den Datacorder eingelegt wird. Spulen Sie mit der [REW] Funktion das Band ganz zurück. Am Ende des Bandes drücken Sie [STOP]. Sollte das Band zufällig aus der Öffnung auf der Vorderseite der Cassette heraushängen, spulen Sie diesen Teil manuell zurück bis es wieder straff ist, sonst könnte das Band und damit Ihre Daten kaputtgehen.

Behandeln Sie die Cassetten vorsichtig. Eine Datencassette ist empfindlicher als eine Toncassette.

Wenn sich zum Beispiel beim Laden das Band verheddert, Sie aber noch laden können, sichern Sie die Daten aus dem Speicher des CPC464 auf eine neue Cassette. Werfen Sie die beschädigte Cassette weg, damit Sie sie nicht aus Versehen wiederverwenden.

2.4 Laden des Welcome-Bandes

Mit dem CPC464 erhalten Sie eine Demonstrationscassette, die eine Reihe von Beispielen zu Graphik- und Musikmöglichkeiten des Computers und seiner eingebauten Software – das Amstrad BASIC und das Betriebssystem – enthält.

Die Software, die den Recorder und die Kommandos zum Lesen und Schreiben steuert, ist ein Teil des Betriebsystems. `LOAD` und `RUN` sind in diesem Teil die häufigsten Kommandos.

Um die Bedienung des Computers so einfach wie möglich zu machen, sind in den CPC464 eine Reihe von Funktionen eingebaut, die Ihnen die Befehlseingabe solange erleichtern sollen, bis Sie sich an die Tastatur gewöhnt haben. Nach Einschalten des Computers meldet er sich mit

```
READY
```

Legen Sie die Cassette, wie in Abbildung 2.1 gezeigt, ein. Setzen Sie den Bandzähler auf 0, indem Sie die schmale Taste neben dem Zählwerk betätigen, dann drücken und halten Sie `[CTRL]` und die kleine `[ENTER]` Taste in dem ausgelagerten Ziffernblock. Folgende Meldung erscheint:

```
RUN"
Press PLAY then any key:
```

(Drücken Sie `[PLAY]` und danach eine beliebige Taste)

Zu Ihrer Information: Dies ist ein Beispiel für die Definition von Funktionstasten. Sie haben darüber in Kapitel 1 kurz gelesen. Es handelt sich dabei um eine `KEY` Anweisung.

Sie hätten auch das Kommando `RUN"` eingeben und `[ENTER]` drücken können, aber mit dem Drücken von 2 Tasten geht es schneller.

In den meisten Fällen haben die beiden `[ENTER]` Tasten die gleiche Funktion. Sie können jedoch die Bedeutung der kleineren der beiden Tasten für bestimmte Zwecke umdefinieren. Über diese 'Frei Definierbaren Befehlstasten' werden Sie noch im Detail informiert.

Die Taste `[PLAY]` finden Sie beim Datacorder. Drücken Sie sie, bis sie einrastet. Die Aufforderung 'eine beliebige Taste' zu betätigen, mag für Unerfahrene eigenartig klingen. Dies ist jedoch allgemein zur Vereinfachung von Funktionsabläufen üblich.

Korrekter wäre es zu sagen, daß Sie irgendeine Taste außer `[SHIFT]`, `[CAPS LOCK]`, `[CTRL]`, `[ESC]` und den Steuertasten des Datacorders drücken sollen. Diese Vereinbarung gilt für dieses Handbuch und alle von Amstrad gelieferten Programme.

Die Taste, die Sie auf diese Rechneranweisung hin drücken, wird nicht auf dem Bildschirm angezeigt. Sie startet den Motor des Datacorders. Sollte das Band nicht starten, versuchen Sie es mit einer anderen Taste (es ist eine gute Angewohnheit, dazu die große [ENTER] Taste zu benutzen). Stellen Sie sicher, daß das Band am Anfang steht und die [PAUSE] Taste nicht gedrückt ist.

Wenn Sie mehr als eine Taste drücken, werden alle folgenden einfach ignoriert, sobald das Laden des Programms begonnen hat.

Beachten Sie, daß Sie keinen Programmnamen angegeben haben. Solange auf derselben Zeile direkt hinter

```
RUN"
```

nichts steht, wird der Computer das erste Programm auf diesem Band suchen und den Ladevorgang beginnen. Wenn der Computer das erste auf diesem Band korrekt aufgenommene Programm gefunden hat, wird folgender Text auf dem Bildschirm erscheinen:

```
Loading WELCOME 1 block 1
```

Diese Meldung sagt Ihnen, daß der erste von mehreren Programmblöcken des Programms *WELCOME 1* geladen wird. Jedes Programm ist in Form von zusammenhängenden Datenblöcken von bis zu 2 KBytes (das entspricht 2048 Zeichen) auf Band gespeichert, die jeder einzeln gekennzeichnet werden. Es wird jeweils ein Block eingelesen und seine Nummer auf dem Bildschirm angezeigt. Nach jedem Datenblock bleibt der Datacorder einen Moment stehen, beginnt dann wieder zu laufen und kurz danach wird die Nummer des neuen Blocks angezeigt.

Sollte der Computer beim Einlesen einen Fehler entdecken, reagiert er mit der auf dem Bildschirm angezeigten Meldung eines Lesefehlers. (Diese sind in Anhang VIII aufgeführt). In diesem Fall bleibt Ihnen nichts anderes übrig, als es wieder und wieder zu versuchen bis der Computer das Programm fehlerfrei lädt.

Lesen Sie die Anweisungen auf dem Bildschirm, während Ihr Programm geladen wird. Ihr Welcome-Band erledigt den Rest.

2.5 Supersafe und Speedload

Der CPC464 bietet zwei Aufnahmegeschwindigkeiten: 'Supersafe' bei einer Übertragungsrate von 1000 Baud (Bits pro Sekunde) und 'Speedload' mit 2000 Baud. Die zweite Geschwindigkeitsstufe ist bei Aufnahmen und Wiedergabe also doppelt so schnell wie die erste, allerdings geht dies auf Kosten der Sicherheitsspanne. Diese kann bei schlechter Qualität billiger Bänder und möglichen Fehlern aufgrund unterschiedlicher Kopfeinstellung bei verschiedenen Recordern notwendig sein.

Für Programme, die mit dem gleichen Gerät aufgenommen und gelesen werden, ist die Speedload-Geschwindigkeit normalerweise sicher genug, solange Cassetten guter Qualität verwendet werden. Auch die meisten gekauften Programme können in dieser Geschwindigkeit ohne Fehler eingelesen werden.

Der Computer erkennt die Aufnahmegeschwindigkeit eines bespielten Bandes und liest in der entsprechenden Geschwindigkeit ein. Wenn Sie mit dem Befehl SAVE ein Programm speichern wollen, müssen Sie dem Computer mitteilen ob Sie die schnellere Speedload-Geschwindigkeit benützen wollen, sonst wird in der langsameren Supersafe-Geschwindigkeit gespeichert.

Wenn Sie Ihr Programm mit 'Speedload' laden wollen, versichern Sie sich zuerst, daß Sie im direkten Modus (auf dem Bildschirm erscheint das Wort Ready) sind und geben dann ein:

```
SPEED WRITE 1
```

Um zur (langsameren) Standardgeschwindigkeit zurückzukehren, können Sie zum einen den Computer komplett zurücksetzen, in diesem Fall gehen allerdings sämtliche Daten und Programme verloren, oder Sie können (im direkten Modus) eingeben:

```
SPEED WRITE 0
```

2.6 Speichern von Programmen und Daten auf Cassette

BASIC bietet Ihnen verschiedene Möglichkeiten, um Daten und Programme auf Cassette zu speichern. Sie werden hier kurz anhand von Beispielen gezeigt.

2.6.1 SAVE "<Dateiname>"

Die einfachste Art der Speicherung geschieht unter Verwendung des Befehls SAVE wenn Sie sich im direkten Modus befinden (nach dem Ausführen oder Auflisten eines Programms erscheint sofort Ready). Nehmen Sie das kleine Beispielprogramm von Kapitel 1, das alle darstellbaren Zeichen aufzeigt.

Der <Dateiname> kann jede Kombination aus maximal 16 Zeichen einschließlich Leerstellen sein. Bei längeren Namen werden die Zeichen ab Stelle 16 ignoriert. Wenn das Programmbeispiel im Speicher des Computers ist, tippen Sie ein:

```
SAVE "ZEICHEN"
```

Der Computer antwortet mit folgender Anweisung:

```
Press REC and PLAY then any key
```

Erinnern Sie sich noch an die Definition von 'irgendeine Taste'? Jetzt wird nach Ausführung der letzten Anweisung das Programm mit dem Namen "ZEICHEN" gespeichert.

WICHTIG!

Wenn [REC] und [PLAY] nicht richtig gedrückt wurden und nur die [PLAY] Taste einrastet, startet zwar das Band, es wird aber nicht beschrieben, obwohl es zunächst so aussieht.

VORSICHT: Wenn Sie aus Versehen zum Einlesen [REC] und [PLAY] drücken wird das Band gelöscht. Sie können, wenn Sie dies bald entdecken, mit [ESC] unterbrechen, aber die Gefahr, daß bereits Daten gelöscht wurden, ist groß. Wenn Sie sich dagegen absichern wollen, auf diese Art Daten zu verlieren, sollten Sie darauf achten, die Laschen der Schreibschutzkerben auszubrechen.

Es gibt vier Möglichkeiten, mit dem CPC464 Dateien zu speichern. Sie haben gerade die übliche Methode gesehen. Drei weitere für Spezialfälle folgen jetzt.

2.6.2 SAVE "<Dateiname>",A

Der Vorgang läuft genauso ab wie oben, bedingt durch die hinzugefügte Information ,A speichert der Computer die Datei nicht in der sonst üblichen Maschinensprache sondern als ASCII Zeichen.

Diese Art Daten zu speichern wird normalerweise in der Textverarbeitung und bei anderen Anwenderprogrammen benützt. Eine detaillierte Beschreibung folgt, wenn diese Programme erklärt werden.

2.6.3 SAVE "<Dateiname>",P

Mit dem Befehl ,P wird der Computer angewiesen, die Daten in codierter Form zu speichern. Dieses Programm kann nicht mehr einfach von jedermann gelesen werden, indem er es mit "RUN" von der Cassette in den Computer lädt und die Ausführung mit [ESC] stoppt. Bevor Sie diese Form der Speicherung wählen, sollten Sie sich für persönliche Zwecke eine ungeschützte Aufzeichnung machen, um sich die Möglichkeit von Änderungen oder einer Herausgabe offenzuhalten.

2.6.4 SAVE "<Dateiname>",B,<Start-adresse>,<Laenge>[,<Anfangspunkt>]

Damit können Sie das Programm, so wie es im RAM gespeichert steht, in binärer Form auf Cassette aufnehmen. Dazu benötigt der Computer folgende Anweisungen: An welcher Stelle im Speicher befinden sich die auf Cassette aufzunehmenden Daten; wie lang ist diese Datensammlung; wie heißt die Speicheradresse, an der die Programmausführung beginnen soll, wenn eine Programmzeile dies befiehlt.

Mit dieser Methode der binären Speicherung können Bildschirmanzeigen direkt auf Cassette gespeichert werden. Damit kann die lange Ladezeit von großen Programmen zum Beispiel durch Einblenden von Bildern oder Titeln aufgelockert werden.

2.6.5 Dateien ohne Namen und CAT (Katalog)

Wenn Sie ein Programm, ohne einen Programmnamen zu verwenden, mit SAVE"" abspeichern, dann wird BASIC das als 'Unbenanntes Programm' speichern. Eine Cassette kann soviele Dateien mit gleichem Namen (oder ohne Namen) aufnehmen wie Platz auf ihr vorhanden ist.

Im Unterschied dazu muß bei Verwendung eines Diskettensystems für jede Datei ein eigener Name verwendet werden.

Daß Sie beim Speichern ohne Namen, die Sie an den Inhalt der Datei erinnern, bald den Überblick über den Cassetteninhalt verlieren, ist klar. Deshalb sollten Sie immer selbsterklärende Namen sowie zusätzliche Hinweise auf Änderungen verwenden.

Sie können den Inhalt einer Cassette `CAT`alogisieren, indem Sie den Befehl `CAT` eingeben und den Anweisungen

```
Press PLAY then any key:
```

folgen. BASIC führt nun jede auf der Cassette gespeicherte Datei auf. Der Dateiname wird in Großbuchstaben angezeigt, gefolgt von der Anzahl der Blöcke des jeweiligen Programms, einem Zeichen, das die Dateiart angibt und dem Wort `ok` wenn die Datei lesbar und damit ladbar ist. Die Zeichen für die Dateiart sind:

$ Standard BASIC Programm
% geschütztes BASIC Programm
* ASCII Datei
& Datei in Binärform

Die Durchführung des `CAT` Befehls hat keinen Einfluß auf das gerade im Speicher befindliche Programm.

2.7 Lesefehler

Wenn Sie die Meldung bekommen, daß ein `Read Error`, ein Lesefehler, während des Ladens vorgekommen ist, läuft das Band zwar weiter und der Computer liest weiterhin die Blöcke, die er nach demjenigen mit dem Lesefehler findet. Er wird sie aber nicht laden, außer er kann sie als Block 1 des Programms identifizieren, das er zuerst schon ohne Erfolg zu laden versucht hatte.

Das heißt, daß Sie nach einem Lesefehler mit `[STOP/EJECT]` das Band stoppen, mit `[REW]` zurückspulen und einfach auf die Taste `[PLAY]` drücken können. Der Computer wird nun wieder versuchen, das Programm zu laden – mit etwas Glück kann dies auch gelingen.

Lesefehler können durch eine Reihe von Gründen verursacht werden. Die häufigsten sind Beschädigungen des Bandmaterials durch mechanische Einwirkungen, zum Beispiel Dehnen oder Zerkratzen. Auch das Ausschalten des Computers, während `[PLAY]`, oder `[REC]` und `[PLAY]`, gedrückt sind, kann zu einem Lesefehler führen.

Denn wenn sich die Stromversorgung entlädt, während das Band am Aufnahme/Wiedergabekopf anliegt, kann der Abschalt-Stromimpuls die ganzen Daten im Bereich des Kopfes zerstören. (Dasselbe gilt für das Einschalten!) Zusätzlich kann durch überlanges Festhalten des Bandes zwischen Capstan-Antrieb und Andruckrolle das Band mechanisch beschädigt werden.

Desgleichen können Lesefehler durch Benutzung der `[PAUSE]` Taste während des Schreib- oder Lesevorgangs entstehen, ebenfalls wenn die Cassette auf einem anderen CPC464 mit unkorrekt eingestelltem Aufnahme/Wiedergabekopf aufgenommen wurde.

Lesefehler können von Zeit zu Zeit auch einfach durch Zufall entstehen. Ursprünglich wurden die Cassetten nicht zur Datenspeicherung sondern zur Tonwiedergabe entwikkelt, und daraus ergeben sich Nachteile gegenüber den wesentlich teureren 'professionellen' Bandspeichersystemen.

Nichtsdestoweniger ist die Cassette ein ausgezeichneter Standard-Datenspeicher zu einem sehr günstigen Preis. Die Größe der Magnetpartikel des Bandes setzt aber zusammen mit der Bandgeschwindigkeit der Schnelligkeit der Datenübertragung zwischen Band und Computer physikalisch bedingte Grenzen. Jeder Versuch, mit Geschwindigkeiten oberhalb der Toleranzgrenze des Systems zu arbeiten, wird daher zu unzuverlässigen Ergebnissen führen – insbesondere bei den Massenvervielfältigungsverfahren, die zur Herstellung von Billigsoftware benützt werden.

ANMERKUNG: Cassetten mit Programmen von anderen Computersystemen können Sie mit dem CPC464 nicht lesen oder laden. Sie mögen genauso aussehen, beim Abspielen mit einem normalen Cassettenrecorder ähnliche Geräusche erzeugen – aber sie können nicht geladen und ausgeführt werden. Sollte Ihnen dies doch gelingen, so bitten wir Sie um eine genaue Angabe des Computertyps und des Programms.

2.8 Überlegungen zu den Cassetten

Obwohl Sie natürlich jeden Cassettentyp (bis zu C90) benützen können, empfehlen wir Ihnen nur C12 (6 Minuten pro Seite) bis höchstens C30 Cassetten. Denn je länger (C60, C90) eine Cassette ist, desto länger dauert es natürlich, bis ein Programm (vorausgesetzt Sie erinnern sich an seinen Dateinamen) vom Bandende gefunden und geladen wird. Eine andere Möglichkeit wäre es, jeweils den Stand des Bandzählwerks in den Dateinamen aufzunehmen. Beim Überschreiben eines Programmes am Ende eines langen Bandes müssen Sie den Anfangspunkt sehr exakt bestimmen und vorsichtig sein, um nicht Teile anderer Programme irrtümlich zu überschreiben.

Alles in allem ist es wohl am besten, mehrere einzelne Cassetten für jeweils nur wenige Programme zu benützen. C12 Cassetten sind relativ billig und deshalb ist auch die Versuchung gering, eine Cassette, die beschädigt, aber noch 'teilweise in Ordnung' ist, zu behalten.

Und bitte denken Sie daran, daß nahezu alle Programme, die sie kaufen können, unter dem Schutz des 'Copyright' stehen. Es ist verboten, auf einer Cassette gespeicherte und gekaufte Software auf irgendeine Art zu vervielfältigen – nicht einmal für einen Freund – es sei denn die Verkaufsbedingungen gestatten dies. (Einige Programme fordern Sie sogar dazu auf, für Sie selbst eine Kopie anzufertigen.) Softwarediebstahl ist strafbar, und die Gesetze werden laufend überarbeitet um jegliches nichtauthorisierte Duplizieren und Handeln mit Software zu unterbinden.

3 BASIC Einführung

Eine kurze Einführung in Programme, die in Amstrad BASIC geschrieben sind.

Themen dieses Kapitels

* Syntax-Regeln und ihre Beschreibung
* PRINT-Befehle, Datenströme und Formatierung der Anzeige
* ZONEn

3.1 BASIC Grundlagen

Der Zusammenhang zwischen der Funktionsweise des CPC464 und dem „eingebauten" BASIC ist in Anhang II beschrieben. Wenn Sie noch nie programmiert haben, so wollen wir Ihnen dabei helfen – aber manchmal wird etwas vorausgesetzt, was dem Anfänger Schwierigkeiten macht. Sollte dies für Sie zutreffen, so lesen Sie am besten ein Buch, das Ihnen einige Grundkenntnisse über Mikrocomputer vermittelt.

Sie sollten sich an Hand der einfachen Beispiele durch das folgende Kapitel arbeiten können; wenn Sie nicht sofort alles verstehen, ist dies auch nicht so schlimm – aber je mehr Regeln Sie lernen, umso leichter wird alles für Sie.

BASIC ist die Sprache, die in Ihren CPC464 'eingebaut' ist. Es ist da, sobald Sie einschalten, und meldet sich sofort mit:

```
Ready
```

BASIC ist die am leichtesten zu erlernende Programmiersprache. Es besteht aus klar definierten Schlüsselwörtern mit einfachen Regeln. Sobald Sie diese verstehen, erscheint Ihnen BASIC vollkommen logisch.

Amstrad BASIC kann die in Kapitel 8 aufgeführten Anweisungen ausführen. Jede Anweisung ist durch eines oder mehrere Schlüsselwörter definiert und kann verschiedene Parameter haben. Diese können wahlfrei sein. Im allgemeinen kann ein Parameter ein Ausdruck mit Konstanten, Variablen und Funktionen sein. Kombinationen von Buchstaben und Ziffern werden 'string' genannt. Verschiedene Formen von Zahlendarstellungen wie dezimal, hexadezimal und binär sind möglich.

Auf Cassetten gespeicherte Daten werden immer sequentiell, das heißt eine nach der anderen, bearbeitet. Im Gegensatz dazu kann bei wahlfreiem Zugriff (Random) eine von vielen Dateien direkt ausgewählt werden, ohne zuerst die davorliegenden und nicht benötigten Dateien lesen zu müssen.

3.2 Die Struktur eines BASIC Programms

Anweisungen werden in BASIC zeilenweise eingegeben. Eine Zeile kann mehrere Anweisungen enthalten, die durch Doppelpunkte getrennt sind. Maximal sind 255 Zeichen pro Zeile möglich. Ein 'Zeichen' ist hier als Buchstabe, Ziffer oder Leerstelle definiert. Im direkten Modus werden Anweisungen direkt mit der Tastatur eingetippt und dürfen keine Zeilennummer enthalten. Im Programm-Modus werden die Anweisungen aus dem Speicher in der Reihenfolge der Zeilennummern durchgeführt.

Amstrad BASIC macht es dem Benutzer möglich, im direkten Modus Zeilen hinzuzufügen oder wegzulassen und bestehende Zeilen zu verbessern. Bevor ein Programm mit `LIST` angezeigt oder mit `RUN` ausgeführt wird, sortiert der Computer die Zeilen entsprechend ihrer Nummer, ganz egal in welcher Reihenfolge sie eingegeben wurden.

3.3 Zeileneingabe

BASIC akzeptiert Zeilen bis zu einer Länge von 255 Zeichen, gefolgt von `[ENTER]`. Während der Eingabe ist es möglich, die im Moment bearbeitete Zeile zu ändern und mit der COPY Cursor Funktion Zeichen aus anderen Zeilen am Bildschirm einzufügen – siehe Kapitel 1.2.7.

Alle Schlüsselwörter müssen mit einem Trennzeichen beendet werden. Dies kann ein Leerzeichen, ein `+`, `-`, oder ein anderes gültiges Trennzeichen sein. Dies ist deshalb nötig, weil Schlüsselwörter oder reservierte Wörter Teile von Variablen sein können – ein reserviertes Wort muß natürlich von anderen Zeichen umgeben sein um als Variable akzeptiert zu werden.

Schlüsselwörter können in Klein- oder mit der `[SHIFT]` Taste in GROSSbuchstaben geschrieben werden.

Die `PRINT` Anweisung kann durch ein `?` ersetzt werden. In diesem Fall ist kein Trennzeichen nötig. Mathematische Zeichen `(+-*/MOD)` beenden ebenfalls Schlüsselwörter. Das folgende Beispiel ist zwar gültig, aber nicht zu empfehlen, da es schlechte Angewohnheiten bei der Programmeingabe fördert. Leerzeichen wären hier besser:

```
for n= 1 to 50:?n:next
```

Genauso kann ein einfaches Anführungszeichen `' ([SHIFT] 7)` die Kommentaranweisung `REM` ersetzen.

Zusätzliche Leerstellen werden nicht berücksichtigt. Sie dienen lediglich der besseren Lesbarkeit eines Programms.

3.4 Sprachregelung

Um BASIC Anweisungen und Schlüsselwörter zu beschreiben, ist eine formelle aber einfache Sprachregelung notwendig. Jede Anweisung ist so beschrieben, wie die Eingabe über die Tastatur erfolgen muß; variable oder wahlfreie Teile werden durch 'Platzhalter' (verschiedene Arten von Klammern) angezeigt. Eine detaillierte Beschreibung folgt im Laufe dieses Abschnitts.

Diese Platzhalter werden durch verschiedene Namen gekennzeichnet und in spitzen Klammern <> dargestellt. An verschiedenen Stellen ist zum Beispiel eine Zahl erforderlich. Dies wird wie folgt dargestellt:

<Zahl>

Alles, was nicht in spitzen Klammern steht, muß exakt wie vorgegeben geschrieben werden. Die Anweisung **STOP** hat zum Beispiel folgende Form:

STOP

Wenn in einer Definition eine wahlfreie Angabe steht, wird dies durch eckige Klammern [] angezeigt; als Beispiel hier eine wahlfrei anzugebende Zahl:

[<Zahl>]

Wenn so eine wahlfreie Angabe mehrfach (das heißt zwischen 0 und einer beliebigen Anzahl) vorkommen kann, wird dies durch ein Sternchen * nach dem Schließen der eckigen Klammer] angezeigt. Eine Ziffernreihe, die aus mindestens einer Ziffer bestehen muß, wird zum Beispiel so dargestellt:

<Ziffer>[<Ziffer>]*

So ein Ausdruck könnte folgendermaßen aussehen;

34
oder 344
oder 345678 etc.

In vielen Fällen werden Angaben innerhalb einer Reihe jeweils durch Kommas getrennt. Die verwendete Kurzform wird am besten durch folgendes Beispiel erklärt:

<Reihe von:<Ausdruck> bedeutet <Ausdruck>[,<Ausdruck>]*
oder
<Reihe von:[#]<Zahl> bedeutet [#]<Zahl>[,[#]<Zahl>]*

Ein Beispiel dafür wäre:

3,4
oder 3,4,4
oder 3,4,5,6,7,8 etc.

So eine Reihe kann auch eine einzelne Angabe sein. Wenn die Reihe mehr als eine Angabe enthält, muß jede zusätzliche Angabe mit einem vorgestellten Komma angefügt werden, denn dieses Zeichen legt die Grenze zwischen vom Computer separat zu bearbeitenden Angaben fest.

Zahlen können in verschiedenen Formen dargestellt werden:

a. <Festpunkt-Zahlen>
   Zahlen ohne Exponent

b. <Gleitpunkt-Zahlen>
   Zahlen, deren Dezimalpunkt verschiebbar ist.
   Die Stelle des Dezimalpunktes wird durch die Angabe eines Exponenten zur Basis 10 festgelegt:
   2E4 (2 mal 10 hoch 4, oder 2.10^4)
   Der Exponent kann positiv sein.

c. Darstellung in hexadezimaler oder binärer Form (siehe Angang II).

Dezimale Form (normale Form)	100	
Hexadezimale Form	&64 oder &H64	(das H ist wahlfrei)
Binäre Form	&X1100100	(das X ist zwingend)

3.5 Übung macht den Meister – Einführung des PRINT Befehls

Zur Demonstration, wie BASIC arbeitet, werden hier einige Beispiele aufgeführt.

Ein Schlüsselwort, das sehr viele Möglichkeiten des Aufbaus von BASIC Anweisungen zeigt, ist PRINT. Ein Kommando ist ein Schlüsselwort, das im direkten oder im Programm-Modus vom Computer ausgeführt wird. Zum Ausführen einer Funktion ist ein Befehl nötig. Ein Beispiel:

```
PRINT FRE("")
```

Damit der CPC464 auf eine Frage antworten kann, müssen Sie ihm drei Dinge sagen:
1. Den Ort, wo die Antwort erscheinen soll – Drucker, Bildschirm oder sonstwo
2. Sie müssen dem Computer die Daten angeben, mit denen er arbeiten soll
3. Sie müssen dem Computer sagen, was er mit den Daten tun soll

Mit PRINT wird der Computer angewiesen, die Ergebnisse eines Befehls auf ein bestimmtes Ausgabegerät (stream) zu schreiben, wobei ein 'stream' durch eine Zahl zwischen 0 und 9 bezeichnet ist. In der BASIC Beschreibung wird dies als <Stream-Ausdruck> (stream expression) bezeichnet. Der Stream-Ausdruck legt fest, welches Ausgabegerät benutzt werden soll:

0...7 sind Textströme, die zu sogenannten Text-'Fenstern' (windows) führen. Diese wurden vorher mit WINDOW definiert.

8 ist der parallele Druckerausgang. Dieser kann nur benutzt werden, wenn ein Drucker mit Centronics-kompatibler Schnittstelle korrekt angeschlossen ist.

9 ist eine Ausgabedatei für den Cassetten-Datenrecorder, die vorher im Programm eröffnet worden sein muß.

Die kurze Form des PRINT Kommandos (ohne Benutzung der PRINT USING Form ist:

PRINT [#<Ein-/Ausgabegerät>][<Ausgabeliste>]

Die eckigen Klammern bedeuten, daß diese Angaben für ein gültiges Kommando nicht unbedingt nötig sind. Wenn Sie nur PRINT eingeben, wird eine leere Zeile am Bildschirm angezeigt – versuchen Sie es einmal. Wenn Sie kein bestimmtes Gerät für die Ausgabe angeben, wählt der CPC464 von sich aus Gerät #0, den Bildschirm hinter dem Cursor. Versuchen Sie dieses Kommando, vergessen Sie dabei nicht, am Ende des Befehls [ENTER] einzugeben, damit der Computer weiß, daß er diesen Befehl jetzt ausführen muß:

```
PRINT "HALLO"
```

Die Antwort ist:

```
HALLO
```

Sie sehen, daß die Anführungsstriche nicht mitgedruckt wurden. Diese doppelten Anführungsstriche werden im BASIC verwendet, um Anfang und Ende einer <Ausgabeliste> zu bestimmen.

Geben Sie jetzt ein:

```
PRINT #Ø,"HALLO"
```

und Sie erhalten das gleiche Ergebnis.

Aber wenn Sie eingeben:

```
PRINT#4,"HALLO"
```

...dann erscheint das Ergebnis ganz oben links am Bildschirm. Dies ist die erste Stelle für das Textfenster 4, das sich standardmäßig über den ganzen Bildschirm erstreckt, wenn Sie nicht vorher mit `WINDOW` etwas anderes festgelegt haben. Die Startposition für jedes Textfenster ist links oben, und 'stream' 4 war noch nicht benutzt. Wenn dieses Kommando jetzt nochmal ausgeführt würde, stünde das zweite `HALLO` unter dem ersten. Dies gilt für jedes Textfenster. Deshalb wurde bei dem Kommando ohne Streamangabe `HALLO` auf die augenblickliche Cursorposition geschrieben.

Diese Funktion von Amstrad BASIC ist besonders mächtig, weil sie den Aufbau von komplexen Bildschirmmasken mit einfachen Kommandos und Definitionen ermöglicht.

BASIC druckt alles aus, was innerhalb der Anführungszeichen steht, ohne den Inhalt zu beachten. Sogar reservierte Wörter sind da erlaubt.

```
PRINT "4*4"
```

die Antwort ist:

```
4*4
```

Wenn Sie wollen, daß BASIC die Rechenoperation ausführt (4 mal 4), müssen die Anführungszeichen weggelassen werden.

```
PRINT 4*4
```

Das Ergebnis ist dann

```
 16
```

Beachten Sie, daß die Zahl eine Spalte links vom linken Rand geschrieben wurde, da BASIC bei einem negativen Ergebnis dort das Minuszeichen drucken würde.

Der `PRINT` Befehl hat noch viele andere Formen, die die vollen Formatierungsmöglichkeiten durch 'Schablonen' nutzen.

Die <Ausgabeliste> im PRINT Kommando bezieht sich auf das, was zu drucken ist. Das kann eine Zahl, eine Variable, ein Zeichenfolgen-Ausdruck (d. h. eine vorher definierte String-Variable, z. B. HALLO$), oder irgendetwas anderes Druckbares in Anführungszeichen sein.

PRINT USING ordnet die Zahlen in ein festes Format, so daß spaltengerecht ausgerichtet wird und Nachkommastellen weggelassen werden können.

3.6 Das PRINT USING Format und ZONEn

Wenn der CPC464 angeschaltet wird, unterteilt BASIC den Bildschirm in Zonen, die jeweils 13 Stellen breit sind (vergleichbar mit einem Tabulator). Wenn die <Ausgabeliste> Kommas zwischen einzelnen Elementen enthält, wird das jeweils nächste Element ab der nächsten Zone gedruckt. Wenn nicht mehr so viele Spalten übrig sind, wie im ZONE-Befehl angegeben ist, wird das nächste Element in eine neue Zeile geschrieben.

Ohne das USING Format druckt BASIC positive Zahlen mit einer Leerstelle, negative Zahlen mit einem Minuszeichen vor der Zahl. Jeder Zahl folgt eine Leerstelle. Der Dezimalpunkt wird nicht gedruckt, wenn kein Wert nach dem Komma steht.

Amstrad BASIC unterstützt die [TAB] Taste nicht als Spaltentabulator, da diese Funktion nicht genügend standardisiert ist und in den verschiedenen BASIC-Versionen unterschiedlich behandelt wird. Wenn man die [TAB] Taste drückt, wird ein Pfeil-Rechts-Zeichen → angezeigt (wie beim gleichzeitigen Drücken der [CTRL] und I-Taste). Sonst hat [TAB] keine weitere Bedeutung im Amstrad BASIC.

3.7 PRINT TAB (<ganzzahliger Ausdruck>) (<Ausgabeliste>)

Die Auswirkung dieses Kommandos kann am besten an einem Beispiel gezeigt werden. Geben Sie dieses Beispielprogramm ein und Sie sehen die Wirkungsweise.

```
5 MODE 2: INK 1,0: INK 0,9
10 FOR N=1 TO 5
20 ZONE 40
30 PRINT TAB(N*4)"HI",N
40 NEXT
```

Dieses Programm zeigt sowohl ZONE zusammen mit Komma, als auch die Funktion von TAB(). Lassen Sie das Programm nochmal laufen, aber ändern Sie vorher Zeile 10 wie folgt:

```
10 FOR N=-5 to 5
```

Die TAB Anweisung bewirkt, daß die Zeichendarstellung erst nach so vielen Leerstellen beginnt, wie in <ganzzahliger Ausdruck> definiert wird. ZONE kann von 1 bis 255 definiert werden, siehe Beschreibung in Kapitel 8.

Das Format PRINT USING wird verwendet, um Zahlen in vernünftig lesbarer Form zu drucken. Es ist etwas schwierig anzuwenden und Sie sollten es an praktischen Beispielen ausprobieren, da seine technische Form

PRINT[#<Ein-/Ausgabegerät>,][<Ausgabe Liste>][<Using-Klausel>]
[<Trennzeichen>]

geradezu als benutzerunfreundlich bezeichnet werden kann, zumal die USING-Klausel weiter unterteilt werden kann in

USING<ganzzahliger Ausdruck>;[<Using-Liste>]

wobei die Using-Liste weiter unterteilt wird in

<Ausdruck>[<Trennzeichen><Ausdruck>]*

Versuchen Sie folgendes Beispiel:

```
PRINT 123.456, USING "###.##";4567.896
```

Das Ergebnis ist dann:

```
123.456     %4567.90
```

Hier wird Verschiedenes demonstriert. Erstens: Das Element vor USING ist nicht betroffen. Zweitens: Die USING Angabe weist die Anzahl der Ausgabepositionen für den Druck zu. Die zu druckende Information kann länger sein als der zugewiesene Platz. Wenn dies links des Kommas zutrifft, wird die Zahl zwar gedruckt, aber ein %-Zeichen weist auf den Überlauf hin. Als nächstes bewirkt das Komma nach 123.456, daß die nachfolgende Zahl ab Beginn der nächsten Druck-ZONE geschrieben wird. Wäre anstelle des Kommas ein Strichpunkt verwendet worden, wäre der Druck nur eine Leerstelle nach 123.456 erfolgt. Zahlen werden aus offensichtlichem Grund immer durch eine Leerstelle getrennt, wenn sie in der gleichen Zeile geschrieben werden.

Beachten Sie schließlich, daß das Ergebnis gerundet ist. Die überzähligen Dezimalstellen werden also nicht einfach ignoriert.

Versuchen Sie dies:

```
PRINT 123.456, USING "#####.##+";4567.899
```

und Sie sehen das Pluszeichen am Ende der formatierten Zahl. Ein Minuszeichen wird bei einer negativen Zahl automatisch gedruckt.

PRINT USING ist eine ausgezeichnete Möglichkeit um Formulare und Listen zu drucken. Sie werden auch immer durch ein % darauf hingewiesen, wenn Überläufe vorkommen.

4 Variablen, Operatoren und Daten

Umgang mit Daten in einem BASIC Programm

Themen dieses Kapitels:

* Typen von Variablen: Real, Integer und Zeichenfolgen (string)
* Operatoren, logische Ausdrücke
* Tabellen
* DATA

4.1 Reservierte Wörter

Sie haben sicherlich schon bemerkt, daß Kommandos und Schlüsselwörter im Amstrad BASIC durch Trennzeichen begrenzt werden, sei es durch Leerstellen, Interpunktionszeichen oder numerische Operatoren usw. Programme sind so leichter zu lesen und auszutesten, weil diese Wörter, selbst wenn Sie mit Kleinbuchstaben eingegeben wurden, beim Auf`LIST`en in Großschreibung aufgezeigt werden. Geschieht dies nicht, dann wurden sie irgendwie falsch eingegeben und das Programm wird nicht richtig laufen.

Bei Amstrad BASIC kann man Schlüsselwörter als Teil von Variablennamen verwenden. Eine Variable ist ein Name, den Sie einem bestimmten Element zuordnen. Das kann ein einzelner Buchstabe sein (Variablen müssen immer mit einem Buchstaben beginnen). Ein langes Programm ist jedoch besser lesbar, wenn Sie selbsterklärende Namen für Ihre Variablen verwenden:

```
ANTWORT=4*4: print ANTWORT
```

In Amstrad BASIC können Variablen bis zu 40 Zeichen lang sein, wobei das erste Zeichen ein Buchstabe sein muß. Leerstellen sind nicht erlaubt. BASIC erkennt sonst die Variable nur bis zur Leerstelle und bringt die Fehlermeldung:

```
Syntax error
```

Dies besagt, daß eine ungültige Folge von Zeichen eingegeben wurde. (Siehe Anhang VIII). Wenn Sie einen mehrere Wörter umfassenden Ausdruck einsetzen möchten, so setzen Sie einen Punkt an Stelle eines Zwischenraumes. Alle gebräuchlichen Formen von indizierten Variablen sind möglich. Denken Sie aber daran, Tabellen mit DIM(ension) zu definieren.

4.2 Kurzschreibweise

Weil es lästig ist, immer wieder PRINT schreiben zu müssen, können Sie dafür ein ? verwenden, und BASIC versteht, daß damit PRINT gemeint ist (solange Sie es nicht innerhalb zweier Anführungszeichen " " schreiben). Beachten Sie, daß hinter dem ? im Unterschied zu PRINT keine Leerstelle folgen muß. Wenn Sie folgende Zeile als Programm schreiben:

```
10?4*4
run
```

ist die Antwort die gleiche. Wenn Sie jetzt das Ein-Zeilen-Programm LISTen:

```
list
10 PRINT 4*4
```

sehen Sie, daß BASIC eine den Regeln entsprechende PRINT Anweisung daraus gemacht hat. In PRINT Anweisungen, in denen Anführungszeichen benutzt werden

```
10? "HALLO"
```

können Sie auch faul sein und das letzte Anführungszeichen in einer Zeile weglassen, so wie bei der Funktion RUN", die durch Drücken von [CTRL] und [ENTER] erzeugt wird. Genauso funktioniert das in Programmzeilen, ist jedoch nicht empfehlenswert wegen der Schwierigkeiten, die beim Verlängern dieser Zeile entstehen können.

4.3 Mehrere Anweisungen in einer Zeile und gemischte arithmetische Ausdrücke

Sie können mehrere Anweisungen in einer BASIC Zeile schreiben. Eine BASIC Zeile kann bis zu 255 Zeichen lang sein und hat nichts mit der Zeilenbegrenzung am Bildschirm zu tun. Die Anweisungen sind voneinander durch : abzugrenzen.

```
?2*8/5+5-4*777E9/3
```

bringt als Antwort

```
-1.036E+12
```

Wichtig ist es, die Reihenfolge zu verstehen, in der die einzelnen mathematischen Ausdrücke (+-*/MOD usw) nacheinander von BASIC abgearbeitet werden. Sie würden sonst nie das richtige Rechenergebnis bekommen. Dies ist die Reihenfolge:

↑ Potenzieren einer Zahl
- Negatives Vorzeichen für eine Zahl
* Multiplizieren
/ Dividieren
\ Division einer Ganzen Zahl: nur der ganzzahlige Wert wird als Ergebnis angezeigt
+ Addition
- Subtraktion

Alles was in Klammern ( ) steht, wird zuerst behandelt. Wenn innerhalb von Klammern wieder gemischte Ausdrücke stehen, dann gelten dafür obige Regeln einschließlich der Klammerregel, falls innerhalb eines Klammerpaares ein weiteres vorhanden sein sollte. Sie müssen die Klammern immer paarweise verwenden, sonst gibt es einen `Syntax Error`.

4.4 Weitere Schritte

Sie haben jetzt schon einiges gelernt, seit Sie in Kapitel 3.5 die `PRINT` Anweisung kennenlernten. Die Grundregeln des Amstrad BASIC, die Sie jetzt kennen sollten, ermöglichen es Ihnen, einen weiteren Schritt in Richtung „echte" Programmierung zu tun. Für die Aufgaben, die wir bisher gemacht haben, hätten wir auch einen einfachen Taschenrechner benutzen können. Sie werden nun mit den nötigen BASIC Schlüsselwörtern vertraut gemacht werden (in Kapitel 8 finden Sie die genaue Beschreibung und Funktionsweise in alphabetischer Reihenfolge).

Viele BASIC Wörter sagen direkt aus, was Sie bedeuten, zumindest in der englischen Sprache. `GOTO 50` heißt: Springe zu Zeile 50 und fahre dort mit der Verarbeitung fort. `END` heißt Ende. BASIC kehrt daraufhin in den Befehlsmodus zurück und zeigt `Ready` an.

Im direkten (oder auch 'unmittelbaren' oder Befehls-) Modus können Sie eine Reihe von Anweisungen eingeben, die durch einen Doppelpunkt `:` getrennt sind. Sobald Sie jedoch `[ENTER]` gedrückt haben, werden diese Anweisungen ausgeführt und diese BASIC Zeile steht dann nicht mehr zur Verfügung. Mit Hilfe der COPY Cursor Technik können Sie jedoch die Anweisungen nocheinmal ausführen lassen, vorausgesetzt die Zeile steht noch am Bildschirm.

4.5 Bedingungen und logische Ausdrücke

BASIC nutzt die Fähigkeit des Computers aus, immer wiederkehrende Aufgaben mit hoher Geschwindigkeit auszuführen – und ohne daß es ihm langweilig wird. Eine Anzahl von Anweisungen ist vorhanden, um diesen Vorgang, die sogenannte Schleifentechnik (looping) zu unterstützen: Beginn, Kontrolle und Beendigung einer Schleife sind somit sehr einfach möglich.

Die letzten dieser Elemente sind die Vergleichsbedingungen. Wie sich verschiedene Daten zueinander verhalten, wird durch 'vergleichen' geklärt. Sie können variable Daten mit anderen Variablen oder mit vorgegebenen, festen Daten vergleichen. Die Vergleichsoperatoren sind:

`<` kleiner als
`<=` kleiner als oder gleich
`=` gleich
`>` größer als
`>=` größer als oder gleich
`<>` ungleich

Jetzt kommt ein kleines Programm, das die Verwendung dieser Operatoren an einem Beispiel, das uns allen am Herzen liegt, zeigt. Wenn Sie nicht schon dieses Bild am Schirm haben:

```
Amstrad   64k Microcomputer (v1)
c 1984 Amstrad Consumer Electronics plc
         and Locomotive Software LTD

BASIC 1.0

Ready
```

dann drücken Sie jetzt [CTRL], [SHIFT] und [ESC]. Sie wissen ja bereits, daß damit der Computer zurückgesetzt wird. Jetzt sehen Sie obiges Bild. Geben Sie nun folgendes Programm ein (Abschnitt 1.2.7 behandelt die Aufbereitung von Programmzeilen):

```
10 INPUT "WIE HOCH IST IHR GEHALT";GEHALT
20 IF GEHALT < 2000 THEN GOTO 30 ELSE 40
30 PRINT "BITTEN SIE UM EINE GEHALTSERHOEHUNG":END
40 PRINT "KAUFEN SIE SICH EIN GROESSERES AUTO"
```

Lassen Sie dieses Programm ablaufen, indem Sie RUN tippen, gefolgt von [ENTER]. Der Computer stellt Ihnen nun folgende Frage:

```
WIE HOCH IST IHR GEHALT?
```

(Beachten Sie, daß der Computer automatisch ein Fragezeichen anfügt, wenn das Programm von Ihnen eine Eingabe erwartet). Geben Sie jetzt nur Ziffern ein, keine Buchstaben, Währungszeichen oder Kommas und beenden Sie wie immer mit [ENTER]

Fügen Sie jetzt die folgende Zeile 5 hinzu, sobald Ready angezeigt wird:

```
5 CLS
```

Führen Sie mit RUN das Programm nochmal aus, um den Schirm zu putzen. War Ihre erste Antwort kleiner als 2000, geben Sie diesmal eine größere Zahl als 2000 ein und Sie sehen den Unterschied. Fügen Sie jetzt nach Ready die Zeile 50 hinzu.

```
5 CLS
10 INPUT "WIE HOCH IST IHR GEHALT";GEHALT
20 IF GEHALT < 2000 THEN GOTO 30 ELSE 40
30 PRINT "BITTEN SIE UM EINE GEHALTSERHOEHUNG": END
40 PRINT "KAUFEN SIE SICH EIN GROESSERES AUTO"
50 IF GEHALT > 4000 THEN PRINT"und machen Sie einen
   schoenen Urlaub"
run
```

Die END Anweisung in Zeile 30 stoppt das Programm. In Zeile 40 ist kein END angegeben, daher fährt das Programm fort und prüft, ob die Gehaltsvariable größer als 4000 ist. In diesem Fall gibt das Programm noch einen weiteren guten Rat.

Machen Sie weiter, und geben Sie noch Zeile 60 ein:

```
5 CLS
10 INPUT "WIE HOCH IST IHR GEHALT";GEHALT
20 IF GEHALT < 2000 THEN GOTO 30 ELSE 40
30 PRINT "BITTEN SIE UM EINE GEHALTSERHOEHUNG":END
40 PRINT "KAUFEN SIE SICH EIN GROESSERES AUTO"
50 IF GEHALT > 4000 THEN PRINT "und machen Sie einen
   schoenen Urlaub"
60 IF GEHALT >3000 THEN PRINT"...und leihen Sie mir
50 DM"
run
```

Sie sehen, daß die Zeichen (> und <) hier als Trennzeichen vor Zahlen benützt werden. Sie brauchen auch keine Leerstelle davor oder danach einzugeben. BASIC würde dies sowieso ignorieren. Wenn Sie als Antwort **3600** eingeben, so sehen Sie, daß die Zeile 50 übersprungen wird, weil die Bedingung nicht zutrifft, aber Zeile 60 ausgeführt wird, da ja hier die Bedingung richtig (wahr) ist.

An dieser Stelle könnten Sie mit den bisherigen Kenntnissen ein eigenes Programm schreiben. Denken Sie daran, wie dieses Programm gewachsen ist. Die meisten Programme enstehen so und hier liegt eine der Stärken von BASIC, weil es dieses Konzept so gut unterstützt.

4.6 Programmentwicklung

Die Art, in der in BASIC Programme entstehen, ist wahrscheinlich sein größter Vorteil. Die, die es ganz genau machen, werden zwar sagen, daß dies zu „unstrukturierten", schlampigen Programmen führt, wenn man immer dann, wenn eine neue Idee vorhanden ist, neue Programmteile hinzufügt. In Wirklichkeit hat das aber große Vorteile. Speziell wenn Sie anfangen zu programmieren und nicht die ganze Übersicht haben, wie das Programm endgültig werden soll. Durch die sofortige Ausführung haben Sie gleich eine Erfolgskontrolle.

Nehmen Sie das Beispielprogramm auf und fügen Sie die Zeile 70 hinzu. Damit wird eine Schleife auf den Programmanfang eingebaut, die nach einiger Zeit, in der Sie die Bildschirmmeldung lesen können, aktiv wird.

```
5 CLS
10 INPUT "WIE HOCH IST IHR GEHALT";GEHALT
20 IF GEHALT < 2000 THEN GOTO 30 ELSE 40
30 PRINT "BITTEN SIE UM EINE GEHALTSERHOEHUNG":END
40 PRINT "KAUFEN SIE SICH EIN GROESSERES AUTO"
50 IF GEHALT > 4000 THEN PRINT "und machen Sie einen
   schoenen Urlaub"
60 IF GEHALT >3000 THEN PRINT "...und leihen Sie mir
   50 DM"
70 for n=1 to 900: next n:goto 5
run
```

Wie Sie sehen, wurden einige Zeilen in Kleinschrift eingegeben. Amstrad BASIC unterscheidet zwischen Schlüsselwörtern und Variablennamen, egal in welcher Schreibweise. Drücken Sie [ESC] zweimal, um das Programm abzubrechen. Mit LIST zeigen Sie jetzt das Programm wieder an. Alle Schlüsselwörter sind jetzt groß geschrieben, nur die Variable n bleibt kleingeschrieben.

Zeile 70 zeigt Ihnen eine Zeitschleife, da das Programm erst von 1 bis 900 zählen muß, bevor die nächste Anweisung ausgeführt wird. Das Programm ist jetzt durch `GOTO 5` in einer Endlosschleife. Sie können das Programm nur durch Drücken der `[ESC]` Taste verlassen. Nach einmaligem Drücken stoppt das Programm. Nach zweimaligem Drücken kehrt BASIC in den direkten Modus zurück. Das Programm ist aber weiterhin im Speicher.

Wenn Sie `[ESC]` drücken während das Programm auf Eingabe wartet, wird es sofort unterbrochen, da es nichts tut. Die Zeilennummer, in der das Programm wartet, wird in der `BREAK` Meldung angezeigt.

```
Break in 10
```

Wenn Sie das Programm während der Warteschleife in Zeile 70 unterbrechen können (zweimal `[ESC]`), kommt die Meldung

```
Break in 70
```

Wenn Sie das laufende Programm in Zeile 70 anhalten, können Sie es ab dieser Stelle weiterlaufen lassen, wenn Sie auf irgendeine Taste drücken. Wenn Sie das Programm ganz abgebrochen haben (zweimal `[ESC]`), so können Sie es trotzdem ab der Abbruchstelle weiterlaufen lassen, wenn Sie

```
CONT
```

eingeben. Was immer Sie auch mit der `[ESC]` Taste tun, das Programm im Speicher geht nicht verloren. Nur wenn Sie `NEW` eingeben oder den Computer zurücksetzen (`[CTRL]`, `[SHIFT]` und `[ESC]`) ist das Programm verloren.

Es ist also kein doppelter Boden notwendig für die, die „versehentlich" die Maschine zurücksetzen. Das Löschen des Programmspeichers geschieht immer vorsätzlich – und gründlich. Sichern Sie alles auf Cassette, was Sie vielleicht wieder brauchen können.

4.7 Weitere Variablen und Zeichenfolgen

Das Arbeiten mit Variablen ist das „Brot in der Suppe" in der Datenverarbeitung. Könnten nur feste Werte verarbeitet werden, wäre der Computer nur eine elektronische Schreibhilfe. Wie Sie wissen, ist das Ergebnis eines mathematischen Ausdrucks variabel, wenn irgendein Teil des Ausdrucks variabel ist. Variable haben drei Eigenschaften:

- Name
- Typ
- „Organisation"

Namen für die Variable wurden in 4.1 behandelt. Der Typ ist wahlweise. Sie können eine Variable nach den Regeln von 3.4 so definieren:

<Name>[<Typenkennzeichen>]

Die Typenkennzeichen sind:

`%` für ganze Zahlen (integer). Nachkommastellen werden ignoriert. Ganzzahlige Variablen brauchen weniger Platz im Speicher. Wenn in einem Programm keine Dezimalstellen verarbeitet werden, können mit `DEFINT` die Variablen ganzzahlig vereinbart werden. Der Wertebereich der ganzen Zahlen ist -32768...+32767.

! definiert eine Variable als Realzahl (*real*). Realzahlen haben Vor- und Nachkommastellen. Nach dem Einschalten des Computers werden alle Variablen als 'real' betrachtet. Deshalb müssen Sie Variable nur dann als 'real' DEFinieren, wenn Sie vorher DEFINT benutzt hatten. Realzahlen können jeden Wert im Bereich 2.9E–39 bis 1.7E+38 annehmen.

$ definiert eine *String-Variable* (Zeichenfolge). Der Inhalt kann jedes beliebige, gültige Zeichen, in Anführungszeichen eingeschlossen, sein.
Z. B.

```
NAME$="DIRK HAGEN"
```

Fügen Sie nun Zeile 6 zu dem Übungsprogramm und ändern Sie Zeile 60:

```
5 CLS
6 INPUT "Wie heissen Sie";NAME$
10 INPUT "WIE HOCH IST IHR GEHALT";GEHALT
20 IF GEHALT < 2000 THEN GOTO 30 ELSE 40
30 PRINT "BITTEN SIE UM EINE GEHALTSERHOEHUNG":END
40 PRINT "KAUFEN SIE SICH EIN GROESSERES AUTO"
50 IF GEHALT > 4000 THEN PRINT
 "und machen Sie einen schoenen Urlaub"
60 IF GEHALT >3000 THEN PRINT
 "...und leihen Sie mir 50 Mark ";NAME$
70 FOR n=1 to 900: NEXT n:GOTO 5
run
```

Eine Leerstelle wurde nach dem Wort Mark eingefügt um zu vermeiden, daß der Name unmittelbar neben Mark gedruckt wird. Der Strichpunkt ; nach den PRINT und INPUT Anweisungen verhindert, daß nach jeder Anweisung eine neue Zeile begonnen wird.

Wir können auch mit ganzen Zahlen (integer) arbeiten. Geben Sie Zeile 61 ein:

```
5 CLS
6 INPUT "Wie heissen Sie";NAME$
10 INPUT "WIE HOCH IST IHR GEHALT";GEHALT
20 IF GEHALT < 2000 THEN GOTO 30 ELSE 40
30 PRINT "BITTEN SIE UM EINE GEHALTSERHOEHUNG":END
40 PRINT "KAUFEN SIE SICH EIN GROESSERES AUTO"
50 IF GEHALT >4001 THEN PRINT
."und machen Sie einen schoenen Urlaub"
60 IF GEHALT >3000 THEN PRINT
 "...und leihen Sie mir 40 Mark";NAME$
61 TAGESGEHALT=GEHALT/30: PRINT
 "das macht DM";TAGESGEHALT;" pro Tag"
70 FOR n=1 to 5000:NEXT n:GOTO 5
run
```

Ändern Sie in Zeile 70 die Zahl 900 in 5000. Das gibt mehr Zeit, um den Bildschirminhalt zu lesen. Das Ergebnis von Zeile 61 ist ungenau. Sie können es zu einer Ganzzahl, (integer) runden. Geben Sie Zeile 62 ein.

```
5 CLS
6 INPUT "Wie heissen Sie";NAME$
10 INPUT "WIE HOCH IST IHR GEHALT";GEHALT
20 IF GEHALT < 2000 THEN GOTO 30 ELSE 40
30 PRINT "BITTEN SIE UM GEHALTSERHOEHUNG": END
40 PRINT "KAUFEN SIE SICH EIN GROESSERES AUTO"
50 IF GEHALT > 4000 THEN PRINT
 "und machen Sie einen schoenen Urlaub"
60 IF GEHALT >3000 THEN PRINT
 "...und leihen Sie mir 50 Mark ";NAME$
61 TAGESGEHALT=GEHALT/30: PRINT
 "das macht DM";TAGESGEHALT;" pro Tag"
62 INTEGER.GEHALT%=TAGESGEHALT:PRINT
"oder DM";INTEGER.GEHALT%;" wenn es Ihnen nicht
 genau auf den Pfennig ankommt"
70 FOR n=1 to 5000: NEXT n:GOTO 5
run
```

Wenn Sie einem Variablennamen ein Typenzeichen zuordnen, müssen Sie diesen im gesamten Programm mit diesem Kennzeichen schreiben, da Sie sonst nicht die gleiche Variable ansprechen. Das Kennzeichen ist Teil des Variablennamens. BASIC Zeilen, die länger sind als eine Bildschirmzeile, werden fortgesetzt. Das passiert in allen 3 Modi. Benutzen Sie `Modus 2` für lange Programme oder überhaupt für Programme, da es viel einfacher ist, zeilenweise zu lesen.

Geben Sie ein:

```
MODE 2
```

für den Modus 2 und wenn Sie mit dem Farbmonitor (CTM644) arbeiten:

```
INK 1,0
INK 0,13
BORDER 13
```

Die schwarze Schrift auf weißem Untergrund ist dann besser zu lesen. Zeigen Sie das Programm jetzt nochmal an (`LIST`).

4.8 Formatieren des Bildschirms

Wenn Sie ein Programm entwickeln, müssen Sie es von Zeit zu Zeit bereinigen. Als erstes sollten Sie die Zeilennummern neu in Zehnerschritten organisieren. Unter der `Ready` Meldung tippen Sie ein

```
RENUM
```

und dann

```
LIST
```

```
10 CLS
20 INPUT "Wie heissen Sie";NAME$
30 INPUT "WIE HOCH IST IHR GEHALT";GEHALT
40 IF GEHALT < 2000 THEN GOTO 50 ELSE 60
50 PRINT "BITTEN SIE UM EINE GEHALTSERHOEHUNG": END
60 PRINT "KAUFEN SIE SICH EIN GROESSERES AUTO"
70 IF GEHALT > 4000 THEN PRINT
 "...und machen Sie einen schoenen Urlaub"
80 IF GEHALT >3000 THEN PRINT
 "...und leihen Sie mir 50 Mark ";NAME$
90 TAGESGEHALT=GEHALT/30: PRINT
 "das macht DM";TAGESGEHALT;" pro Tag"
100 INTEGER.GEHALT%=TAGESGEHALT: PRINT
 "oder DM";INTEGER.GEHALT%;" wenn es Ihnen nicht
 genau auf den Pfennig ankommt"
110 FOR n=1 to 5000: NEXT n:GOTO 10
```

Alle Zeilennummern sind jetzt in Zehnerschritten geordnet. Das Wichtigste dabei ist, daß auch die Verweise im Programm auf die Zeilennummern berichtigt wurden. Jetzt sollten Sie die Bildschirmausgaben überarbeiten.

Dazu schalten Sie zuerst die Zeitschleife in Zeile 110 aus, indem Sie aus dem Befehl einen Kommentar (**REM**ark) machen:

```
10 CLS
20 INPUT "Wie heissen Sie";NAME$
30 INPUT "WIE HOCH IST IHR GEHALT";GEHALT
40 IF GEHALT < 2000 THEN GOTO 50 ELSE 60
50 PRINT "BITTEN SIE UM EINE GEHALTSERHOEHUNG": END
60 PRINT "KAUFEN SIE SICH EIN GROESSERES AUTO"
70 IF GEHALT > 4000 THEN PRINT
 "und machen Sie einen schoenen Urlaub"
80 IF GEHALT >3000 THEN PRINT
 "...und leihen Sie mir 50 Mark ";NAME$
90 TAGESGEHALT=GEHALT/30: PRINT
 "das macht DM";TAGESGEHALT;" pro TAG"
100 INTEGER.GEHALT%=TAGESGEHALT: PRINT
 "oder DM";INTEGER.GEHALT%;"wenn es Ihnen nicht
 genau auf den Pfennig ankommt"
110 REM FOR n=1 to 5000: NEXT n:GOTO 10
```

Wenn man **REM** an den Anfang der Zeile setzt, wird der Rest der Zeile von BASIC übergangen. Bemerkungszeilen dienen hauptsächlich der besseren Lesbarkeit eines Programmes. Da das Programm jetzt nach Zeile 110 stehenbleibt und BASIC sich mit **READY** meldet, bleibt der Bildschirminhalt erhalten. Geben Sie jetzt ein:

```
15 mode 1
```

Zeile 15 bestimmt jetzt den Darstellungsmodus, und welcher Modus auch immer gesetzt war, Zeile 15 setzt den Modus 1. Das **MODE** Kommando löscht außerdem den Bildschirm, weil automatisch ein **CLS** durchgeführt wird.

Zeile 10 ist damit überflüssig. Lassen Sie sie aber trotzdem jetzt stehen.

Lassen Sie jetzt das Programm ablaufen und geben Sie Antworten ein:

```
Wie heissen Sie? Hans
WIE HOCH IST IHR GEHALT? 4250
KAUFEN SIE SICH EIN GROESSERES AUTO
und machen Sie einen schoenen Urlaub
...und leihen Sie mir 50 Mark Hans
das macht DM 141,666667 pro Tag
oder DM 142
wenn es Ihnen nicht genau auf den Pfennig ankommt
Ready
```

Das ist nicht sehr elegant, besonders weil das Wort "Pfennig" getrennt wird. Fügen Sie

```
25 PRINT: PRINT
85 PRINT
```

hinzu und ändern sie (EDIT) 100 wie folgt:

```
100 INTEGER.GEHALT%=TAGESGEHALT: PRINT
"oder DM";INTEGER.GEHALT%;"" Wenn es Ihnen":PRINT
"nicht genau auf den Pfennig ankommt"
```

Wenn Sie jetzt das Programm ausführen, sehen Sie, wie schön der Text gedruckt wird (in Modus 1). Mit Zeile 120 rücken Sie die READY Meldung weiter nach unten.

```
120 ?:?:?:?:?
```

Lassen Sie das Programm noch einmal laufen oder geben Sie ein:

```
120 GOTO 120
```

Jetzt kommen Sie nur über [ESC] aus dem Programm. Denken Sie daran, ? ist die Kurzform für PRINT. LISTen Sie das Programm und prüfen Sie das Ergebnis.

```
10 CLS
15 MODE 1
20 INPUT "Wie heissen Sie";NAME$
25 PRINT:PRINT
30 INPUT "WIE HOCH IST IHR GEHALT";GEHALT
40 IF GEHALT < 2000 THEN GOTO 50 ELSE 60
50 PRINT "BITTEN SIE UM EINE GEHALTSERHOEHUNG": END
60 PRINT "KAUFEN SIE SICH EIN GROESSERES AUTO"
70 IF GEHALT > 4000 THEN PRINT
   "und machen Sie einen schoenen Urlaub"
80 IF GEHALT >3000 THEN PRINT
   "...und leihen Sie mir 50 Mark";NAME$
85 PRINT
90 TAGESGEHALT=GEHALT/30: PRINT
   "das macht DM";TAGESGEHALT;" pro Tag"
100 INTEGER.GEHALT%=TAGESGEHALT: PRINT
    "oder DM";INTEGER.GEHALT%;" wenn es Ihnen ":PRINT
    "nicht genau auf den Pfennig ankommt"
110 REM:FOR n=1 to 5000: NEXT n:GOTO 10
120 GOTO 120
```

4.9 LOCATE

Die bis jetzt eingesetzten BASIC Kommandos entsprechen dem allgemein üblichen Standard, so wie BASIC auf den meisten Maschinen eingesetzt ist. `LOCATE` ist eine Besonderheit von **Amstrad** BASIC (und einigen anderen BASIC Dialekten) zur wahlfreien Positionierung des Cursors auf dem Bildschirm:

```
LOCATE 10,4
```

bringt den Cursor in die vierte Zeile von oben und die zehnte Spalte von links. Wenn Sie das im direkten Modus tun, wird der Cursor zwar in die richtige Position gebracht, aber durch die `Ready` Meldung sofort auf eine neue Zeile und an den linken Rand gebracht. Fügen Sie Zeile 16 hinzu:

```
16 LOCATE 10,4
run
```

Sie sehen, daß die erste Anfrage richtig eingeordnet wird. Die nächste Zeile wird wie vorher behandelt, da Zeile `25` einige leere Zeilen einfügt. Drücken Sie zweimal `[ESC]` und fügen Sie Zeile `26` hinzu:

```
26 LOCATE 10,4
```

Wenn Sie jetzt das Programm ausführen, werden beide Fragen in Zeile 4 Spalte 10 angezeigt, wobei die zweite Frage die erste überschreibt. Fahren Sie nun fort, den Text am Bildschirm zu positionieren. Sie können dazu die Bildschirm-Koordinaten benutzen, die in Anhang VI aufgeführt sind.

Wollen Sie alle Fragen und Kommentare an der gleichen Stelle anzeigen, sollten Sie `CLS:` vor jedem `LOCATE` angeben, damit die neue Anzeige nicht in den Resten der vorhergehenden Texte untergeht.

Die Koordinaten von `LOCATE` können auch Variablen sein, gefüllt mit Werten aus dem Programm. Das 'Gehalt'-Programm hat nun seinen Dienst getan. Wollen Sie das Programm behalten, so speichern Sie es auf Cassette (siehe dazu Kapitel 2). Vielleicht wollen Sie es mit den Befehlen erweitern, die jetzt eingeführt werden.

4.10 IF... THEN

Der Sinn und die Funktionsweise entsprechen ganz genau dem Sprachgebrauch von WENN... DANN. `IF` <logischer Ausdruck> `THEN GOTO` <Zeilennummer> ist eine von mehreren Möglichkeiten dieses Kommandos.

`IF` prüft, ob die Bedingung des <logischen Ausdrucks> erfüllt wird. In diesem Fall wird die Option ausgeführt. Die `IF` Anweisung kann in Schleifen eingebaut werden, die wiederholt abgearbeitet werden (sog. Zeitschleifen – recursive loops). Setzen Sie den Computer mit `[CTRL], [SHIFT]` und `[ESC]` zurück und tippen Sie ein:

```
1 MODE 1
10 AMSTRAD=0
20 PRINT "AMSTRAD CPC464 colour personal
 computer"
30 AMSTRAD = AMSTRAD +1
40 IF AMSTRAD <10 GOTO 20
```

Führen Sie dieses Programm aus. Das Programm läuft solange in einer Schleife, bis die Bedingung in Zeile 40 erfüllt ist. Zeile 40 schleift also zurück auf 20. Sie können sehen, was der Begriff 'Variable' meint, denn mit jeder Schleife ändert sich der Wert von Amstrad

Wenn Sie sehen wollen, wie sich der Wert von `Amstrad` im Programm ändert, so können Sie dies mit der Anweisung in Zeile 35 verfolgen.

```
35 LOCATE 1,20: PRINT AMSTRAD:LOCATE 1,AMSTRAD
run
```

Wenn dies zu schnell war, bauen Sie eine Warteschleife ein:

```
36 for n=1 to 500:next
```

Jetzt geben Sie, wenn Sie einen `CTM644` Farbmonitor haben, ein bißchen Farbe dazu:

```
34 BORDER AMSTRAD
```

Damit wird die Farbe durch den Wert der Variablen `AMSTRAD` gesetzt und bei jeder Darstellung geändert. `LIST`en Sie das Programm wieder auf:

```
1 MODE 1
10 AMSTRAD=0
20 PRINT "AMSTRAD CPC464 colour personal
 computer"
30 AMSTRAD = AMSTRAD +1
34 BORDER AMSTRAD
35  LOCATE 1,20:PRINT AMSTRAD:LOCATE 1,AMSTRAD
36 FOR n=1 TO 500:NEXT
40 IF AMSTRAD <10 GOTO 20
run
```

Sie wollen nun natürlich alle Farben sehen und ändern deswegen Zeile 40...

```
40 IF AMSTRAD <26 GOTO 20
```

Wenn Sie das Programm jetzt laufen lassen, sehen Sie von den vorhandenen 26 Farben zuerst die dunkelste und dann die hellste. Mit der folgenden Änderung können Sie die Meldung etwas sinnvoller gestalten:

```
35 LOCATE 1,20:PRINT "Border ";AMSTRAD:
 LOCATE 1,AMSTRAD
run
```

Weil Sie gerade bei den Farben sind, geben Sie nach Programmende (`Ready` Meldung) ein:

```
BORDER 14,6
```

und Sie werden sehen, daß die Farbe des Bildrandes zwischen 14 und 6 wechselt. Im nächsten Kapitel erfahren Sie alles über Graphik und Farbe. Haben Sie bis dahin noch etwas Geduld. Um diesen Abschnitt etwas abzurunden, geben Sie folgendes ein:

```
ink 1,18,16
```

dann...

```
speed ink 1,5
```

Jetzt haben Sie eine Pause verdient. Wenn Sie Ihre Freunde beeindrucken wollen, dann speichern Sie das Programm auf Cassette. Setzen Sie den Computer zurück.

4.11 Tabellen

Manche Programme brauchen eine Menge Variablen um Daten zu speichern. Das ist an und für sich kein Problem, aber es kann schwierig werden, die Variablen und deren Inhalt im Griff zu behalten. Glücklicherweise hat BASIC dafür eine Lösung in Form von Tabellen (Arrays).

Sie werden fragen: Was ist eine Tabelle? Das ist eine Gruppe von Variablen, die alle mit dem gleichen Namen angesprochen werden.

Am besten ist das anhand eines Beispiels zu verstehen. Denken Sie an ein Programm, das ein Kartenspiel simuliert. Da werden wahllos Karten gezogen. Also muß über jede Karte Buch geführt werden. Ein einfacher Weg dies zu tun, wäre es, jeder Karte eine Variable zuzuordnen und diese nach dem Aufenthaltsort der Karte auf einen bestimmten Wert zu setzen – sagen wir 1 wenn sie gespielt wurde, und 0 wenn Sie noch nicht gespielt wurde.

Daraus folgt, daß Sie 52 Variablen (für jede Karte eine) brauchen und sich erinnern müssen, welche Variable zu welcher Karte gehört. Hier beginnt der Nutzen einer Tabelle.

Zuerst geben Sie der Tabelle einen Namen, sagen wir `STAPEL`. Um ein Element einer Tabelle anzusprechen, geben Sie ihm einfach eine Nummer. Wenn die ersten 13 Elemente Herz repräsentieren, dann könnte Herz ASS gleich `STAPEL (1)`, König gleich `STAPEL (2)` und Herz 6 gleich `STAPEL (9)` sein.
Sehen Sie das Prinzip?

Unglücklicherweise können Sie nicht unbegrenzt Variable definieren. Sie müssen dem Computer eine Möglichkeit geben, Platz im Speicher freizuhalten. Dies geschieht mit der Anweisung `DIM`.

Mit diesem Befehl wird die `DIM`ension einer Tabelle festgelegt. In Ihrem Beispiel sind das 4 mal 13 Karten, das macht 52 Variable.

Die Anweisung sieht damit so aus:

```
DIM STAPEL (52)
```

damit weiß BASIC, daß Platz für 52 Elemente im Speicher reserviert werden muß (tatsächlich werden 53 reserviert, da auch Element 0 vorhanden ist).

Sie können jetzt mit den Vorarbeiten für ein Unterprogramm zum Kartenspielen beginnen:

```
10 DIM STAPEL(52)
20 FOR X = 1 TO 52
30 LET STAPEL(X)=0
40 NEXT X
......
1000 KARTE = INT(RND*52)+1
1010 IF STAPEL(KARTE) = 1 THEN GOTO 1000
1020 STAPEL(KARTE) = 1
1030 RETURN
```

Beachten Sie, daß die **DIM** Anweisung die erste Zeile des Programms ist. Dies ist deswegen notwendig, weil eine Tabelle nur einmal im Programm definiert und nicht umdefiniert werden kann.

In den Zeilen 20 bis 40 wird den Elementen der Tabelle der Wert 0 zugeordnet. Das Unterprogramm, das in Zeile 1000 startet, sucht wahllos eine Karte aus und prüft, ob sie bereits gespielt wurde. Wenn ja, wird solange eine andere gesucht bis eine gefunden ist, die noch nicht gespielt wurde. Der Wert dieses Elements wird dann auf 1 gesetzt, um anzuzeigen, daß die Karte gespielt ist. Dann wird das Unterprogramm bei **RETURN** verlassen und das Hauptprogramm fährt fort.

Die Tabellenbearbeitung ist nicht auf einstufige Tabellen beschränkt. Solange Platz im Speicher ist, können Sie soviele Stellen verwenden, wie Sie wollen. Eine dreistufige Tabelle mit je 10 Elementen würde so definiert:

```
DIM TABELLE (10,10,10)
```

Diese Technik ist sinnvoll, um große Datenbestände in kleinere übersichtliche Einheiten zu bringen. In unserem Beispiel mit dem Kartenspiel könnte man auch 4 Einheiten mit je 13 Elementen definieren, die dann einzeln zugänglich sind. Dann kann man jede Karte anhand ihrer Farbe bestimmen:

```
DIM STAPEL (4,13)
```

Wenn Sie jetzt zum Beispiel die Kreuz 4 suchen (in der ersten Tabelle war das Element 43), dann müssen Sie jetzt **STAPEL (2,4)** heraussuchen – unter der Voraussetzung, daß Kreuz als zweite 'Reihe' der Tabelle definiert war.

In Tabellen können Zahlen wie auch Zeichenfolgen gespeichert sein, zum Beispiel die Namen der Besucher in einem Theater pro Sitz und Reihe.

4.12. DATA Anweisung

Diese Anweisung kann zusammen mit der Anweisung **READ** verwendet werden, um Daten in ein Programm einzugeben. Die benötigten Daten stehen in Programmzeilen durch Kommas getrennt und durch eine **DATA** Anweisung am Anfang definiert. Nun können diese Daten sequentiell, d. h. nacheinander mit einer **READ** Anweisung gelesen werden.

Ein Beispiel dafür:

```
10 READ X,Y,Z
20 PRINT X;"+";Y;"+";Z;" = ";X+Y+Z
30 DATA 1,3,5
```

Die Daten können entweder Zahlen, algebraische Ausdrücke oder Zeichenfolgen sein. Wenn nicht alle Zeichen in eine Zeile passen, schreiben Sie einfach eine neue, wieder mit dem Wort **DATA** am Anfang. Wenn der Computer auf eine **READ** Anweisung stößt, durchsucht er das ganze Programm nach den nächsten Daten, ganz egal, wo diese stehen. Sorgen Sie dafür, daß zu jeder **READ** Anweisung Daten vorhanden sind, sonst treten Fehler auf.

Die einzige Möglichkeit, den **READ** Vorgang zu unterbrechen, ist die Anwendung des **RESTORE** Kommandos. Damit wird der Cursor auf die erste **DATA** Anweisung am Programmanfang zurückgesetzt. So kann man diese Daten wenn nötig mehrere Male lesen. Das folgende Programm demonstriert die Benutzung der **DATA**, **READ** und **RESTORE** Kommandos:

```
10 FOR C =1 TO 5
20 READ X$
30 PRINT X$;" ";
40 NEXT C
50 RESTORE
60 GOTO 10
70 DATA HALLO,WIE,GEHT'S,DIR,HEUTE
```

Mit **[ESC]** kommen Sie aus diesem Programm heraus.

Beachten Sie, daß die **DATA** Anweisung zwar hier am Ende des Programms steht, sie kann aber an jedem Platz stehen.

Die **DATA** Anweisung müssen Sie nicht unbedingt nur dazu benutzen, um Daten zu definieren, die mit **PRINT** ausgegeben werden sollen. Numerische Werte können auch z. B. in einem **SOUND** Befehl eingelesen werden.

Geben Sie ein:

```
10 FOR n=1 TO 30
20 READ s
30 SOUND 1,s,40,5
40 NEXT n
50 DATA 100,90,100,110,120,110,100,0
60 DATA 130,120,110,0,120,110,100,0
70 DATA 100,90,100,110,120,110,100,0
80 DATA 130,0,100,0,120,150
```

Wenn Sie nichts hören, stellen Sie bitte die Lautstärke mit dem rechts am Computer angebrachten Regler ein.

Zum Abschluß dieser kurzen Einführung in BASIC ist hier ein Programm, mit dem Sie „17 und 4“ (englisch: Pontoon) spielen können. Es demonstriert die Anwendung von vielen BASIC Funktionen. Aufgrund entsprechender Variablennamen sollte es leicht verständlich sein. Sie können kreativ sein und Graphik- und Klang-Funktionen hinzufügen und so aus diesem „Basis“-BASIC-Programm ein Meisterwerk machen.
Das Ziel dieses Spieles ist es, möglichst nahe an die Punktesumme 21 heranzukommen. Sollten Sie über 21 kommen, so sind Sie ausgeschieden. Wenn Sie das Progrmm eintippen, geben Sie nach Zeile 1 den Befehl `AUTO` ein, dies generiert automatisch die folgenden Zeilennummern.

```
1 REM PONTOON
10 REM INITIALISIEREN
20 YC=2:CC=2
30 ASSE=0
40 CASSE=0
50 S=0
60 T=0
70 DIM FARBE$(4)
80 FARBE$(1)="KREUZ"
90 FARBE$(2)="HERZ"
100 FARBE$(3)="PIK"
110 FARBE$(4)="KARO"
120 CLS
130 DIM STAPEL(52)
140 FOR X=L TO 52
150 STAPEL(X)=0
160 NEXT X
170 REM JEDEM SPIELER ZWEI KARTEN GEBEN
180 LOCATE 10,3
190 PRINT"SIE";SPC(15)"BANK"
200 LOCATE 3,5
210 GOSUB 740
220 S=S+F
230 IF F=11 THEN ASSE=ASSE+1
240 LOCATE 3,6
250 GOSUB 740
260 S=S+F
270 IF F=11 THEN ASSE=ASSE+1
280 LOCATE 24,5
290 GOSUB 740
300 T=T+F
310 IF F=11 THEN CASSE=CASSE+1
320 LOCATE 24,6
330 GOSUB 740
340 T=T+F
350 IF F=11 THEN CASSE=CASSE+1
360 REM EINGABE: KARTE (G)EBEN/KARTEN (A)UFLEGEN
370 X$=INKEY$:IF X$<>"g" AND X$<>"a" THEN 370
380 IF X$="S" THEN 560
390 LOCATE 3,YC+5
400 YC=YC+1
```

```
410 GOSUB 740
420 S=S+F
430 IF F=11 THEN ASSE=ASSE+1
440 REM PUNKTE UND ASSE
450 IF S<22 THEN 3670
460 IF ASSE=0 THEN 500
470 ASSE=ASSE-1
480 S=S-10
490 GOTO 450
500 LOCATE 12,19
510 PRINT "SIE SIND PLEITE"
520 PRINT:PRINT"EIN NEUES SPIEL (J/N)?"
530 9$=INKEY$: IF 9$<>"j" AND 9$<>"n" THEN GOTO 530
540 IF 9$="j" THEN RUN
550 END
560 IF T>16 THEN 700
570 CC=CC+1
580 LOCATE 24,CC+4
590 GOSUB 740
600 T=T+F
610 IF F=11 THEN CASSE=CASSE-1
620 IF T<21 THEN 560
630 IF CASSE=0 THEN 670
640 CASSE = CASSE-1
650 T=T-10
660 GOTO 620
670 LOCATE 12,19
680 PRINT "SIE HABEN GEWONNEN!"
690 GOTO 520
700 LOCATE 12,19
710 IF T<S THEN 680
720 PRINT"DIE BANK HAT GEWONNEN!"
730 GOTO 520
740 REM KARTE GEBEN
750 LET KARTE=INT(RND(1)*52+1)
760 IF STAPEL(KARTE)=1 THEN 750
770 STAPEL(KARTE)=1
780 F=KARTE-13*INT(KARTE/13)
790 IF F=0 THEN F=13
800 IF F=1 OR F>10 THEN 850
810 PRINT F ;" ";
820 IF F>10 THEN F=10
830 PRINT FARBE$(INT((KARTE-1)/13)+1)
840 RETURN
850 IF F=11 THEN PRINT "BUBE ";
860 IF F=12 THEN PRINT "DAME ";
870 IF F=13 THEN PRINT "KOENIG ";
880 IF F<>1 THEN GOTO 820
890 F=11
900 PRINT "ASS ";
910 GOTO 830
```

Geben Sie G ein für 'Geben' (die nächste Karte, die Sie bekommen), oder A, um aufzulegen. Wir behaupten nicht, daß dies das beste Programm für den CPC464 sein muß. Es ist eine gute Grundlage, die Sie mit Graphik und Musik entsprechend ausbauen können.

4.13 Logische Ausdrücke

Ein grundlegender Unterschied zwischen einem Computer und einer Rechenmaschine ist die Fähigkeit des Computers, logische Ausdrücke wie IF...THEN bearbeiten zu können. Die Art und Weise wie dies geschieht, ist in dieser kurzen Definition schwierig zu beschreiben. Die Daten, die mit diesen logischen Ausdrücken verarbeitet werden, werden intern als BitMuster dargestellt und Bit für Bit einzeln abgearbeitet.

Die zwei Hälften eines logischen Ausdrucks werden Argumente genannt. Ein logischer Ausdruck umfaßt:
<Argument>[<Vergleichsoperator><Argument>]

wobei
<Argument> ist: NICHT <Argument>
oder: <Numerischer Ausdruck>
oder: <Bezugsbedingung>
oder: (<Logischer Ausdruck>)

Beide Argumente für einen Vergleichsoperator müssen ganze Zahlen sein. ERROR 6 wird angezeigt, wenn dies nicht der Fall ist.

Die logischen Operatoren in der Reihenfolge ihrer Bearbeitung und ihre Auswirkung auf jedes Bit sind:

AND Ergebnis ist 0 wenn nicht beide Argumentbits 1 sind
OR Ergebnis ist 1 wenn nicht beide Argumentbits 0 sind
XOR Ergebnis ist 1 wenn nicht beide Argumentbits gleich sind

AND ist der meistbenutzte Operator und bedeutet **nicht** dasselbe wie 'addieren'.

```
PRINT 10 AND 10
```

Führt zum Ergebnis 10

```
PRINT 10 AND 12
```

Führt zum Ergebnis 8

```
PRINT 10 AND 1000
```

Führt wieder zum Ergebnis 8.

Das kommt daher, weil beide Zahlen, 10 und 1000, in ihre Binärdarstellung umgewandelt wurden:

```
      1010
1111101000
```

Dann wird Bit für Bit verglichen. Wenn in beiden Zahlen an gleicher Position eine 1 gefunden wird, ist die Antwort 1:

```
0000001000
```

Wieder in Dezimalform umgewandelt, ist das Ergebnis wieder 8. Das bedeutet nichts anderes, als daß **AND** prüft, ob zwei Bedingungen gleichzeitig erfüllt sind. Hier haben Sie ein selbsterklärendes Beispiel:

```
10 CLS:INPUT "Der wievielte Tag des Monats ist
   heute";Tag
20 INPUT "Der wievielte Monat";Monat
30 IF Tag=24 AND Monat=12 GOTO 50
40 GOTO 10
50 PRINT "Frohe Weihnachten!"
```

OR vergleicht ebenfalls Bits. Das Ergebnis ist immer 1, außer wenn beide Bits 0 sind. In diesem Fall ist das Ergebnis 0. Nehmen wir die gleichen Zahlen wie beim **AND** Beispiel:

```
PRINT 1000 OR 10
 1002
```

Bitmuster

```
      1010
1111101000
```

Das Ergebnis ist:

```
1111100010
```

Und im Programmbeispiel:

```
10 CLS:
20 INPUT "Der wievielte Monat";Monat
30 IF Monat=12 OR Monat=1 OR Monat=2 GOTO 50
40 CLS:GOTO 10
50 "Jetzt ist Winter"
```

```
10 CLS
20 INPUT "Der wievielte Monat";Monat
30 IF NOT(Monat=6 OR Monat=7 OR Monat=8) goto 50
40 CLS:GOTO 10
50 PRINT "Jetzt ist nicht Sommer"
```

Bedenken Sie, daß Sie mehrere logische Bedingungen miteinander verknüpfen können, um zu einer klaren Entscheidung zu kommen (bis zur maximalen Zeilenlänge):

```
10 CLS:INPUT "Der wievielte Tag des Monats ist
   heute";Tag
20 INPUT "Der wievielte Monat";Monat
30 IF NOT(Monat=12 OR Monat=1) AND Tag=29 GOTO 50
40 CLS:GOTO 10
50 PRINT "Es ist weder Dezember noch Januar, aber
   es koennte ein Schaltjahr sein"
```

Das Ergebnis eines Vergleichsausdrucks ist entweder −1 oder 0. Die Bitdarstellung für −1: Alle Bits der Ganzzahl sind 1; für 0 sind alle Bits 0. Das Ergebnis einer logischen Vergleichsoperation bei zwei solchen Argumenten ergibt −1 für 'Wahr' (`True`) oder 0 für 'Falsch' (`False`).

Prüfen Sie dies, indem Sie dem Programm Zeile 60 hinzufügen:

```
60 PRINT NOT(Monat=12 OR Monat=1)
70 PRINT (Monat=12 OR Monat=1)
```

Wenn das Programm läuft, geben Sie 29 für den Tag und 5 für den Monat ein. Dann wird die Meldung aus Zeile 50 angezeigt und die Ergebnisse des Vergleichs werden in Zeile 60 und 70 darunter angezeigt.

Schließlich ist das Ergebnis von `XOR` (e`X`clusives `OR`) Wahr, wenn beide Argumente unterschiedlich sind. Die folgende Tabelle zeigt alle Möglichkeiten, die bei logischen Vergleichen existieren. Dies ist eine gute Demonstration dessen, was bei bitweisen logischen Operationen passiert.

Argument A	1	0	1	0
Argument B	0	1	1	0
`AND` Ergebnis	0	0	1	0
`OR` Ergebnis	1	1	1	0
`XOR` Ergebnis	1	1	0	0

5 Graphik Einführung

Einführung in Farben und Graphik des CPC464

Themen dieses Kapitels

* Bildschirm-Modus und Bildpunkte
* Farben
* INK, PAPER und PEN
* Zeichnen von Linien
* Fenster (Windows)

5.1 Maschinenspezifische Eigenschaften

Amstrad BASIC entspricht zum großen Teil dem Industriestandard von BASIC. Die meisten Kommandos sind ohne größere Änderungen auf andere BASIC Versionen übertragbar. Die Kommandos für Graphik, Farbe und den Text-Cursor sind speziell darauf ausgerichtet, wie die CPC464-Hardware den Bildschirm steuert und kontrolliert. Nur wenn Sie diese Kommandos wirklich verstehen, können Sie alle Möglichkeiten des CPC464 ausnutzen.

Wie üblich sind die BASIC Schlüsselwörter in der Schrifttype geschrieben, wie sie am Bildschirm verwendet wird. Beispiele und Beschreibung der Schlüsselwörter und ihrer Funktionen finden Sie in Kapitel 8.

5.1.1 Farbe

Schwarz, das heißt keine Farbe oder Anzeige, wird im folgenden immer als Farbe betrachtet wenn es um die Beschreibung von Farbattributen und der verschiedenen Kommandos geht, die sich mit Farbe beschäftigen.

`BORDER`, der Bildrand, kann unabhängig vom Bildschirm Modus auf jedes Farbpaar eingestellt werden. Ein `MODE`-Kommando hat keine Auswirkung auf `BORDER`.

Die Anzeige kann zwischen zwei Farben alternieren oder unverändert in einer Farbe erfolgen.

Die Anzahl der verfügbaren Schriftfarben (`INK`), die gleichzeitig verwendet werden können, hängt ab vom Bildschirm Modus, der mit `MODE` bestimmt wurde. `INK` kann entweder alternierend mit zwei Farben oder gleichbleibend mit einer Farbe verwendet werden. Die Farbe von Text-Papier (`PAPER`), Stift (`PEN`) und Graphik-Stift kann aus der vorhandenen `INK` Palette gewählt werden.

5.1.2 Transparent-Darstellung und die Beziehungen von PEN, INK und PAPER

Außer wenn alternierende Farben gewählt sind, werden 2 Farben (INK) zum Beschreiben des Bildschirms verwendet. Eine Farbe stellt die Schrift (PEN), eine andere den Untergrund (PAPER) dar.
Achtung: Die Zahl im PAPER-Kommando gibt INK an, das mit dieser Zahl definiert ist und NICHT die Nummer der Farbe, die im Anhang IV angegeben ist. Genauso verhält es sich beim PEN Kommando. Auch da definiert die Zahl beim PEN Kommando INK und nicht die Nummer der Farbe, wie sie im Anhang IV aufgeführt ist.

Wenn keine Nummer für PAPER angegeben wird, ist der Standard-Wert 0, während bei PEN ohne Zahlenangabe der Standard-Wert 1 ist. Um INK auf grün für PAPER 0 zu setzen, was der Farbe Nummer 9 entspricht, geben Sie folgende Anweisung ein:

```
INK 0,9
```

Ähnlich verhält es sich, wenn Sie INK schwarz, das ist Farbnummer 0, für PEN 1 setzen wollen. Tippen Sie:

```
INK 1,0
```

Wenn Sie PAPER und PEN mit der gleichen INK belegen, z. B. INK 0,0 wird die ganze Anzeige schwarz.

Der Text kann transparent gemacht werden (der Cursor ist immer transparent) oder undurchsichtig. Benutzen Sie dazu die vorhandenen Steuerzeichen, die die Möglichkeiten der BASIC Graphik-Kommandos noch beträchtlich erweitern. Im transparenten Modus können Sie entweder die PAPER Farbe ignorieren und die Graphik überlagern, oder den gesamten Hintergrund überschreiben. Dieses kurze Beispielprogramm demonstriert die vorhandenen Möglichkeiten:

```
[CTRL][SHIFT][ESC]

10 MODE 1
20 INK 2,19
30 DRAW 200,200,2
40 LOCATE 1,21
50 PRINT "1 NORMAL"
60 PRINT CHR$(22)+CHR$(1)
70 ORIGIN 0,0
80 DRAW 500,200,2
90 LOCATE 12,18
100 PRINT"2 TRANSPARENT"
110 PRINT CHR$(22)+CHR$(0)
120 LOCATE 22,15
130 PRINT"3 WIEDER NORMAL"
```

Der erste DRAW-Befehl in Zeile 30 wird durchgeführt, bevor der transparente Modus mit Kommando CHR$(22)+CHR$(1) in Zeile 60 aktiviert wird. Sie sehen, wie die sich überlagernden Punkte in der Farbe geändert wurden und wie im transparenten Modus die Tinte (INK), die die Buchstaben füllt, komplett überlagert wurde.

Tauschen Sie die Zeilen 60 und 110 und beachten Sie, wie sich die Anzeige ändert. Sie finden im Anhang VI die komplette Liste dieser zusätzlichen Kommandos.

5.2 Bildschirm-Modus

Es gibt drei Modi für den Bildschirm. Das gilt für Text und Graphik.

a) Standard

Modus 1: 40 Spalten/Zeile, 25 Zeilen, 4 `INK` für Text,
320 x 200 Bildpunkte, 4 Farben.

b) Vielfarben Modus

Modus 0: 20 Spalten/Zeile, 25 Zeilen, 16 `INK` für Text,
160 x 200 Bildpunkte, 16 Farben.

c) Hochauflösend

Modus 2: 80 Spalten/Zeile, 25 Zeilen, 2 `INK` für Text,
640 x 200 Bildpunkte, 2 Farben.

Wie Sie sehen können, liegt der Unterschied in der Anzahl der horizontalen Bildschirmelemente. Lassen Sie sich nicht durch die Streifen am Bildschirm verwirren, die Sie bei genügender Helligkeit der Anzeige sehen können. Das ist ganz normal bei einem Bildschirm.

Jede dieser drei verschiedenen Anzeigearten wird als Anzeige-Modus bezeichnet. Nur einer kann jeweils aktiv sein. Wenn Sie von einem Modus auf einen anderen umschalten, wird die ganze Bildschirmanzeige einschließlich aller Text- und Graphik-Fenster gelöscht (siehe `CLS` und `CLG`). Das Programm im Speicher wird nicht berührt.

Sie können den Modus im Programm oder direkt ändern.

5.2.1 `MODE 0` Vielfarben Graphik Modus

16 der 27 Farben können gleichzeitig angezeigt werden. Jedes einzelne Bildschirmelement kann mit einer Farbe programmiert werden. Jede Zeile hat 160, jede Spalte hat 200 Bildpunkte. Im Anhang VI finden Sie einen Plan dieses Rasters.

Im `MODE 0` können Sie 20 Zeichen/Zeile in 25 Zeilen darstellen.

5.2.2 `MODE 1` Standard Modus

Wenn Sie den CPC464 einschalten, ist `MODE 1` aktiviert. 4 aus 27 Farben können gleichzeitig angezeigt werden. Sie können, wenn Sie wollen, sehr schnell zwischen den 27 Farben umschalten. Jede Zeile hat 320, jede Spalte hat 200 Bildpunkte.
Auch dieses Raster finden Sie im Anhang VI.

Im `MODE 1` stehen 40 Zeichen/Zeile in 25 Zeilen zur Verfügung.

5.2.3 `MODE 2` Hochauflösend

Im `MODE 2` können gleichzeitig 2 Farben angezeigt werden. Er wird hauptsächlich verwendet, wenn man 80 Zeichen Text pro Zeile erzeugen will. Damit eignet sich dieser

Modus speziell zum Erstellen von Programmen, weil man hier einen viel größeren Programmausschnitt sehen kann.

MODE 2 liefert je Zeile 640 und je Spalte 200 Bildpunkte.

5.2.4 Versuchen Sie dies...

Setzen Sie den CPC464 vollständig zurück mit [CTRL],[SHIFT] und [ESC]. Geben Sie folgendes Programm ein:

```
5 REM GRAPHIK BEISPIEL
10 MODE 1
15 INK 2,0
16 INK 3,6: REM SETZE FARBE FUER DRAW IN 90
17 BORDER 1: REM DUNKEL BLAU
20 CLG: REM LOESCHE DIE ANZEIGE
30 b%=RND*5+1 :REM SETZE PSEUDO
 RANDOM INTEGER VARIABLE
40 c%=RND*5+1
50 ORIGIN 320,200 :REM LEGE DEN GRAPHIK-NULLPUNKT
 FEST
60 FOR a = 0 TO 1000 STEP PI/30
70 x%=100*COS(a)
80 MOVE x%,x% :REM VERSCHIEBE DEN GRAPHIK CURSOR
90 DRAW 200*COS(a/b%),200*SIN(a/c%),3
 :REM ZEICHNE DIE LINIE
91 IF INKEY$<>""THEN 20
100 NEXT :REM ZURUECK NACH 60 WENN KEINE UNTER-
 BRECHUNG IN 91
110 GOTO 20
```

Mit RUN starten Sie dieses Programm. Drücken Sie irgendeine Taste, wenn Sie ein neues Muster erzeugen wollen. Dieses Programm zeigt einige wichtige Eigenschaften der Hard- und Software des CPC464: Der CPC464 beschreibt den Bildschirm sehr weich und gleichmäßig ohne Flackern und Flimmern, und die Software kennt Befehle, mit denen auf einfache Art und Weise ganz tolle Effekte erzielt werden können. Die Anweisung REM (Bemerkung, REMark) gibt Ihnen die Möglichkeit, Ihr Programm zu dokumentieren. Damit können Sie und Andere verstehen, wie das Programm arbeitet.

Anhand der Zeilennummer kann man erkennen, welche Anweisungen nachträglich eingefügt wurden. Mit RENUM werden die Zeilen neu in Zehnersprüngen numeriert. Sie können aber auch die Originalnumerierung belassen, um zu sehen, wie das Programm schrittweise entstanden ist.

Speichern Sie nun mit SAVE dieses Programm auf Cassette, zum Beispiel:

```
SAVE "GRAPHIK 5.5.84"
```

Dieses Beispiel zeichnet ein mehrfarbiges Muster:

```
new
10 a$=INKEY$: REM DRUECKE IRGENDEINE TASTE UM EIN
 NEUES MUSTER ZU ERZEUGEN
20 IF a$="" THEN 10
30 CLS
40 m=INT(RND*3):REM ERZEUGT ZUFALLSZAHL
 ZWISCHEN 0 UND 3
50 IF m>2 THEN 40:REM NEUER VERSUCH WENN
 DER WERT 2 UEBERSTEIGT
60 MODE m
70 i1=RND*26:REM ZUFAELLIGE INK AUSWAEHLEN
80 i2=RND*26
90 IF ABS(i1-i2)<5 THEN 70
100 INK 0,i1:INK 1,i2
110 s=RND*5+3
120 ORIGIN 320,-100
130 FOR x= -1000 TO 0 STEP s
140 MOVE 0,0
150 DRAW x,300:DRAW 0,600
160 MOVE 0,0
170 DRAW -x,300: DRAW 0,600
180 a$=INKEY$
190 IF a$<>"" THEN 30:REM UNTERBRECHEN DER
 SCHLEIFE DURCH BELIEBIGEN TASTENDRUCK
200 NEXT x
210 GOTO 10
```

Dieses und das vorherige Beispiel zeigen sehr farbig und bildhaft mathematische Begriffe. Beide Programme tun im Grunde das Gleiche: Durch einen Zufallsgenerator werden willkürlich Zahlen erzeugt, die nach entsprechender Behandlung unterschiedliche Linienmuster am Bildschirm ausgeben.

Ihr CPC464 ist ein ideales 'elektronisches Zeichenbrett'. Eines der klassischen geometrischen Muster ist eine Sinuskurve:

```
10 REM SINUSKURVE ZEICHNEN
20 MODE 2
30 INK 1,21
40 INK 0,0
50 CLS
60 DEG
70 ORIGIN 0,200
80 FOR n=0 TO 720
90 y=SIN(n)
100 PLOT n*640/720,198*y,1
110 NEXT
```

Die Anweisung **PLOT** in Zeile 100 zeichnet die Linie Punkt für Punkt nach jedem Rechenergebnis der **FOR NEXT** Schleife (Zeilen 80 – 110).

Der CPC464 hat viele einfache und mächtige Kommandos. Sie können den Effekt dieses Programms erhöhen, wenn Sie folgendes hinzufügen:

```
15 BORDER 6,9
```

Lassen Sie das Programm erneut ablaufen. Der Bildrand (border) wechselt jetzt zwischen den Farben 6 und 9. Die Schnelligkeit des Wechsels ist durch die Standardwerte vorgegeben. Damit das Programm ununterbrochen in Schleife solange läuft, bis Sie [ESC] drücken (einmal um das Programm anzuhalten, zweimal um es abzubrechen), fügen Sie folgende Zeile hinzu:

```
120 GOTO 50
```

Der Bildrand wechselt seine Farbe immer noch, auch wenn das Programm gestoppt ist, weil die Randfarbe unabhängig vom Programm gesteuert wird. Sie können das Blinken des Programmes stoppen und die Farbe in ein helles Blau ändern, wenn Sie zweimal [ESC] drücken und Zeile 15 folgendermaßen ändern:

```
15 BORDER 2
```

Wenn Sie jetzt das Programm ausführen, hört das Blinken auf. Sie können die Farbe der Kurve und des Hintergrundes ändern, indem Sie die Farben von INK in Zeile 30 und 40 ändern. Mit LIST sollten Sie jetzt folgende Programmliste bekommen:

```
10 REM SINUSKURVE ZEICHNEN
15 BORDER 2
20 MODE 2
30 INK 1,2
40 INK 0,20
50 CLS
60 DEG
70 ORIGIN 0,200
80 FOR n=0 TO 720
90 y=SIN(n)
100 PLOT n*640/720,198*y,1
110 NEXT
120 GOTO 50
```

Die Zahl 1 am Ende der PLOT-Anweisung befiehlt dem Computer, die Kurve in der Farbe der INK 1 Anweisung in Zeile 30 zu zeichnen. Lesen Sie in Kapitel 8 den genauen Aufbau und die Funktionsweise des PLOT-Kommandos nach.

Wenn Sie die Kurve, die am Bildschirm gezeichnet wird, genau ansehen, werden Sie bemerken, daß es keine stetige Linie ist, sondern lauter kleine Punkte, die gegeneinander verschoben sind. Diese Punkte sind die Bildpunkte (Pixel), die bereits früher behandelt wurden.

5.2.5 Der Graphikcursor und das Zeichnen von Linien

Sie haben jetzt verschiedene Wege ausprobiert, Graphik per Programm zu erstellen. Sie konnten einige Kommandos und Regeln testen. Beim Zeichnen von Linien am Bildschirm müssen Sie einige wichtige Punkte beachten, um nicht verwirrt zu werden.

Der erste Punkt, den es zu beachten gilt, ist der aktuelle Zustand des Programmspeichers. Die Farben bleiben aktiv, auch wenn ein Programm beendet ist. Selbst wenn Sie mit `NEW` ein neues Programm beginnen oder einfach damit das alte löschen, hat das keine Auswirkung auf die augenblicklichen Farben. Nur wenn Sie den Computer zurücksetzen, haben Sie einen wirklich leeren Speicher. (Sie sollten vorher alles, was Sie nochmal brauchen können, sichern!)

Überzeugen Sie sich jetzt:

```
NEW:CLS
```

nachdem Sie das vorherige Programm abgebrochen haben. Tippen Sie ein:

```
DRAW 100,100
```

Die `DRAW`-Anweisung zieht eine Linie von der letzten Graphik-Cursor Position zu der x,y Koordinate der `DRAW` Anweisung (100,100). Der Graphik-Cursor ist unsichtbar und steht immer an der Position, an der die Graphik Anweisung etwas zeichnet.

Wenn Sie die Position finden wollen, müssen Sie die Funktionen `XPOS` und `YPOS` verwenden. Geben Sie jetzt ein:

```
PRINT XPOS
```

Die Anwort ist

```
100
```

(das gleiche gilt an diesem Punkt auch für `YPOS`)

Wenn der Text die letzte Bildschirmzeile überschreitet und dadurch das Bild nach oben gerollt wird, rollt auch die Graphik nach oben, aber der Graphik-Cursor behält seine Position. Probieren Sie es. Drücken Sie ↓ bis das Bild nach oben verschwunden ist. Testen Sie dann den Wert von `XPOS` und `YPOS`. Er hat sich nicht verändert.

Wollen Sie in einer bestimmten Farbe zeichnen, so hängen Sie diese Anweisung an das Ende des `DRAW`-Befehls an (genauso wie beim `PLOT`-Befehl auf der vorherigen Seite).

Zuerst müssen Sie aber `INK` spezifiziert haben. `INK` ist limitiert durch den Modus des Bildschirms, der gerade aktiv ist.
Folgendes Programm zeigt Ihnen das:

```
10 MODE 1
20 INK 0,10
30 ORIGIN 0,0
40 INK 1,26
50 INK 2,0
60 DRAW 320,400,1
70 DRAW 640,0,2
```

Das folgende Programm demonstriert all die Möglichkeiten, die bisher besprochen wurden und zeigt ein paar nützliche mehr. In Zeile 10 werden INK und Farbe so definiert, daß alle möglichen Überreste des letzten Programms gelöscht sind und das neue Programm die erwünschten Ergebnisse liefert:

```
10 INK 0,0:INK 1,26:INK 2,6:INK 3,18: BORDER 0
20 REM Dieses Programm zeichnet Muster
30 mode 1:DEG
40 PRINT "3,4 oder 6seitige Muster ? ";
50 LINE INPUT p$
60 IF p$="3" THEN sa=120:GOTO 100
70 IF p$="4" THEN sa=135:GOTO 100
80 IF p$="6" THEN sa=150:GOTO 100
90 GOTO 50
100 PRINT:PRINT "Ich rechne";
105 IF p$="3" THEN ORIGIN 0,-50,0,640,0,400
 ELSE ORIGIN 0,0,0,640,0,400
110 DIM cx(5),cy(5),r(5),lc(5)
120 DIM np(5)
130 DIM px%(5,81),py%(5,81)
140 st=1
150 cx(1)=320:cy(1)=200:r(1)=80
160 FOR st=1 TO 4
170 r(st+1)=r(st)/2
180 NEXT st
190 FOR st=1 TO 5
200  lc(st)=0:np(st)=0
210  np(st)=np(st)+1
220  px%(st,np(st))=r(st)*SIN(lc(st))
230  py%(st,np(st))=r(st)*COS(lc(st))
240  lc(st)=lc(st)+360/r(st)
245  IF (lc(st) MOD 60)=0 THEN PRINT ".";
250  IF lc(st) < 360 THEN 210
252  px%(st,np(st)+1)=px%(st,1)
254  py%(st,np(st)+1)=py%(st,1)
260 NEXT st
265 CLS:ink 1,2
270 st=1
280 GOSUB 340
290 LOCATE 1,1
300 EVERY 25,1 GOSUB 510
310 EVERY 15,2 GOSUB 550
320 EVERY 5,3 GOSUB 590
330 GOTO 330
340 REM Kreis zeichnen und 3,4 oder 6 andere
    um diesen
350 cx%=cx(st):cy%=cy(st):lc(st)=0
360 FOR x%=1 TO np(st)
370 MOVE cx%,cy%
380 DRAW cx%+px%(st,x%),cy%+py%(st,x%),
1+(st MOD 3)
390 DRAW cx%+px%(st,x%+1),cy%+py%(st,x%+1),
1+(st MOD 3)
400 NEXT x%
```

```
410 IF st=5 THEN RETURN
420 lc(st)=0
430 cx(st+1)=cx(st)+1.5*r(st)*SIN(sa+lc(st))
440 cy(st+1)=cy(st)+1.5*r(st)*COS(sa+lc(st))
450 st=st+1
460 GOSUB 340
470 st=st-1
480 lc(st)=lc(st)+2*sa
490 IF (lc(st) MOD 360)<>0 THEN 430
500 RETURN
510 ik(1)=1+RND*25
520 IF ik(1)=ik(2) OR ik(1)=ik(3) THEN 510
530 INK 1,ik(1)
540 RETURN
550 ik(2)=1+RND*25
560 IF ik(2)=ik(1) OR ik(2)=ik(3) THEN 550
570 INK 2,ik(2)
580 RETURN
590 ik(3)=1+RND*25
600 IF ik(3)=ik(1) OR ik(3)=ik(2) THEN 590
610 INK 3,ik(3)
620 RETURN
```

Beim Ablauf dieses Programmes werden Sie in Zeile 40 nach einer Eingabe gefragt. Antworten Sie mit 3. Das bringt das schnellste Ergebnis. Dann erscheint die Meldung `"Ich rechne"` und alle paar Sekunden (Zeile 245) wird ein Punkt angezeigt, der darauf hinweist, daß das Programm noch „nachdenkt".

In den Zeilen 300 bis 320 werden Unterprogramme aufgerufen. Dort werden die verschiedenfarbigen `INK` zum Blinken gebracht. Den Takt liefern die `EVERY` Anweisungen. Wenn Sie den Ablauf langsamer machen wollen, ändern Sie die Zeilen 300–320 wie folgt:

```
300 EVERY 250,1 GOSUB 510
310 EVERY 150,2 GOSUB 550
320 EVERY 50,3 GOSUB 590
```

Die Beschreibung von `EVERY` finden Sie im Kapitel 8. Das ist eine der nützlichsten Eigenschaften von Amstrad BASIC. Sehen Sie einmal, was passiert, wenn Sie das Programm mit `[ESC]` anhalten und dann ein paar Sekunden später durch Drücken irgendeiner Taste wieder starten.

Das Programm versucht die verlorene Zeit aufzuholen. Die Anzeige blinkt so schnell wie möglich, da die Anweisungen gespeichert wurden. Da dem Programm nur eine beschränkte Menge Speicher zur Verfügung steht, wird, wenn der Speicher voll ist, die nächste `EVERY` Anweisung ignoriert, bis wieder Platz ist. Platz wird geschaffen, wenn man das Programm weiterlaufen läßt und die gespeicherten Anweisungen abgearbeitet werden können.

5.3 Fenster (Windows)

Sie können den Bildschirm in bis zu 8 Textfenster einteilen, in die Texte geschrieben werden können, und in ein Graphikfenster. Fenster werden durch die `MODE` Anweisung auf Ihre Standard-Werte zurückgesetzt. In Kapitel 8 finden Sie eine nähere Beschreibung.

Beachte: Wenn das Textfenster den ganzen Bildschirm umfaßt (das ist die Voreinstellung beim Rücksetzen) sorgt die Hardware für schnelles 'Durchrollen'. Wenn das Textfenster nicht den ganzen Bildschirm umfaßt, übernimmt die Software das Rollen, und das ist entsprechend langsamer.

Die `WINDOW`-Anweisung spezifiziert die Fenstergrenzen des entsprechenden Bildschirm-'Streams'. Fenster können sich überlappen. Damit haben Sie eine schnelle Methode, ausgefüllte Rechtecke zu zeichnen. Bevor Sie das jetzt untersuchen, geben Sie ein:

```
KEY 139, "mode 2:paper 0:ink 1,0:ink 0,9:
          list"+chr$(13)
```

Damit wird die kleine `[ENTER]` Taste als Funktionstaste definiert, mit dem Zweck, den Text in sichtbare Farben zu bringen wenn Sie sich mit `PEN` und `PAPER` mal selbst ausgetrickst haben, indem Sie beiden die gleiche Farbe zugeordnet haben und damit nichts mehr lesen können. Das folgende Beispiel zeichnet einige Fenster über den ganzen Bildschirm und zeigt zwei wichtige Punkte:

```
5 MODE 0
10 FOR n=0 TO 7
20 WINDOW #n,n+1,n+6,n+1,n+6
30 PAPER #n,n+4
40 CLS #n
50 FOR c=1 TO 200:NEXT
60 NEXT
```

Der erste besagt, daß jedes neue Fenster ein altes überschreibt, der zweite, daß Meldungen immer auf stream `#0` gehen, wenn man nicht ausdrücklich etwas anderes vereinbart. Bevor Sie irgendetwas anderes tun, tippen Sie:

```
LIST
```

Das Programm wird auf `Stream 0` ausgegeben. Versuchen Sie:

```
LIST #5
```

Dann:

```
CLS #6
```

Dies zeigt Ihnen, daß der letzte Bildschirm-'Stream' alle anderen überschreibt bis auf die System Meldung `READY`, die über `Stream 0` kommt, auch wenn das Listing nach `Stream 5` geschickt wurde. Fügen Sie Zeile 55 hinzu um die `WINDOW SWAP` Anweisung zu verwenden:

```
55 IF n=3 THEN WINDOW SWAP 7,0
```

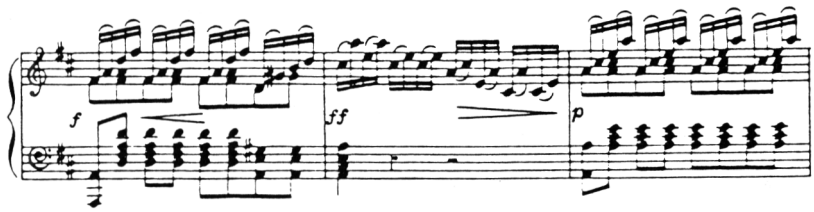
Sie werden zu Recht vermuten, daß diese Anweisung die `READY` Meldung in das Fenster mit der Nummer 7 ausgibt. Mit diesem einfachen Programm haben Sie eine Vorstellung bekommen, wie `WINDOW`s funktionieren.

6 Musik-Einführung

Klangeffekte werden durch einen Lautsprecher im Computer erzeugt. Wenn Sie den Modulator MP2 und Ihren Fernseher verwenden, dann stellen Sie die Lautstärke des Fernsehers ganz leise.

Die Lautstärke regeln Sie mit dem `VOLUME` Regler ganz rechts außen am Computer. Sie können den Ton auch auf den Auxiliary-Anschluß Ihres Stereo-Gerätes geben, wenn Sie diesen mit dem `I/O` Anschluß links an der Computerrückseite verbinden. So können Sie die Musik, die vom Computer erzeugt wird, über Ihre HiFi-Lautsprecher oder Kopfhörer anhören.

Der Klang des CPC464 ist hervorragend. Sie müssen allerdings die Philosophie, die hinter der Klangsoftware steckt, verstehen, um zum bestmöglichen Ergebnis zu kommen.



Themen dieses Kapitels:

* Tonperioden
* Musik-Befehle
* Hüllkurven
* Speichern und Synchronisieren

Wenn Sie nur möchten, daß Ihr Computer einen „Piepston" von sich gibt, dann geben Sie ein:

```
PRINT CHR$(7)
```

und überspringen den Rest dieses Kapitels. Aber dann werden Sie einige der interessantesten und lohnendsten Eigenschaften des CPC464 nie kennenlernen. Dieses Kapitel gibt zuerst einen Gesamtüberblick über die Klangeigenschaften, und erklärt dann ausführlich die Schlüsselwörter und deren Anwendung. Sie sollen in die Lage versetzt werden, Klangfolgen zu erzeugen, verschiedene Instrumente nachzubilden, und damit Melodien zu spielen.

6.1 Grundsätzliches zu `SOUND`

Wenn Sie den Klang einer Note in einer Melodie hören, werden Sie verschiedene Eigenschaften bemerken:

1) Tonhöhe und Änderung der Tonhöhe während der Dauer einer Note.
2) Lautstärke und Änderung der Lautstärke während der Dauer einer Note.
3) Länge einer Note.

6.2 Tonhöhe

Bei einer Note ist die Tonhöhe der wichtigste Faktor. Eine Musiknote kann als eine regelmäßige Schwingung beschrieben werden. Alle Schwingungen haben eine Frequenz, eine Periode und eine Amplitude (siehe Bild 1).

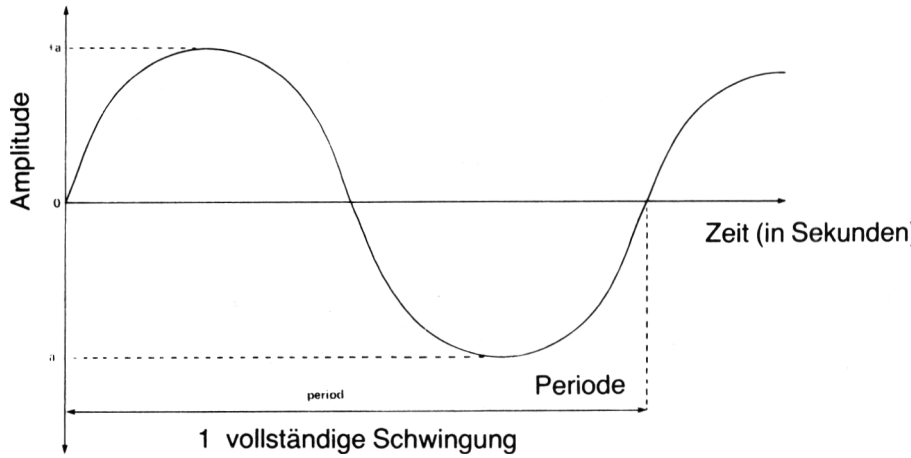


Bild 1: Die Charakteristiken einer Note

Die Frequenz ist die Anzahl der Schwingungen pro Sekunde. Die Periode ist die Zeit für eine ganze Schwingung. Die Amplitude ist eine Eigenschaft der Lautstärke und spielt hier für die Definition der Tonhöhe einer Note keine Rolle.

Die Beziehung zwischen Frequenz und Periode kann durch folgende Formel ausgedrückt werden:

Frequenz *(in Hz)* = 125000/<Tonperiode>

Somit erzeugt eine <Tonperiode> von 1000 einen 125 Hz Ton, und eine <Tonperiode> von 125 erzeugt einen Ton von 1000 Hz (1kHz).

Auf beide Arten könnte man die Tonhöhe definieren. Für Amstrad BASIC wurde die <Tonperiode> gewählt. Lassen Sie sich nicht durch die Tatsache verwirren, daß der Wert für die Tonhöhe sinkt, wenn der Wert für die Periode steigt.

Die Tonhöhe kann in der **SOUND**-Anweisung über den Parameter <Tonperiode> gesetzt werden. Der Bereich für Tonperioden geht von 0 bis 4095 und muß als Ganzzahl dargestellt werden. Das kann gelegentlich zu Rundungsfehlern führen, wenn Sie eine Melodie aufbauen, aber selbst das geschulteste Ohr wird dies kaum wahrnehmen (siehe Anhang VII).

Wenn eine Note auf einem echten Musikinstrument gespielt wird, kann sich die Tonhöhe ändern, manchmal ist das auch Absicht (Vibrato). Mit der **ENT**-Anweisung (**EN**velope Ton, Hüllkurve für den Ton) kann eine spezielle Form für die Struktur des Periodenwechsels während der Dauer einer jeden Note festgelegt und im **SOUND**-Befehl verwendet werden. Diese Technik wird im folgenden als die <Tonhüllkurve> bezeichnet.

6.3 Lautstärke

Die Lautstärke ist eigentlich das Maß dafür, wie laut der Ton ist. Um den Anfangswert der Lautstärke in einem **SOUND**-Kommando zu setzen, steht Ihnen eine Skala zur Verfügung:

Wählen Sie aus dem Bereich 0 – 15 eine Zahl, und zwar eine Ganzzahl (integer). Je höher die Zahl, desto lauter ist der Ton. Dieser Wert ist ebenfalls Bestandteil des **SOUND**-Kommandos und wird dort als <Lautstärke> bezeichnet. Wenn Sie bereits einige einfache Noten ausprobiert haben, ist Ihnen sicher aufgefallen, wie langweilig eine normale Note klingt, wenn sie nicht 'behandelt' wird. Das kommt daher, daß bei einem herkömmlichen Musikinstrument der Ton zunächst anschwillt und dann abklingt.

Die verschiedenen Formen des Anschwellens und Abklingens für die verschiedenen Musikinstrumente können mit dem Computer über die elektronische Lautstärkeregelung simuliert werden. Das geschieht mit der **ENV**-Anweisung und wird als <Lautstärken-Hüllkurve> bezeichnet.

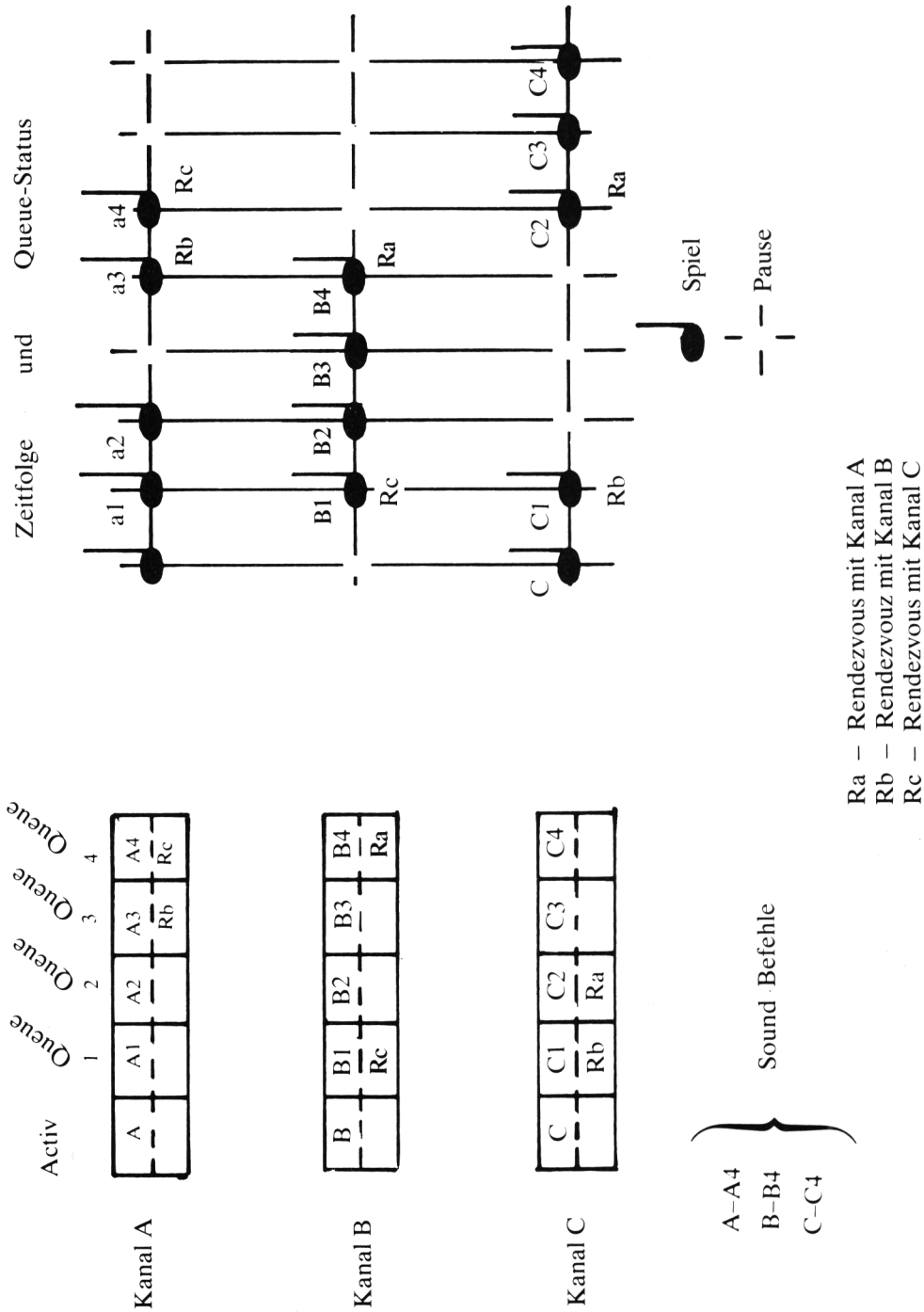
6.4 Länge einer Note

Die Notenlänge ist ein wesentlicher Bestandteil eines jeden Musikstückes. Die Grundeinheit der musikalischen Notenlänge ist die Viertelnote.
Es gibt Achtel-, Halb- und ganze Noten. Immer sind dies Teile oder Vielfache der Viertelnote. Melodien können jedoch in verschiedenen Geschwindigkeiten gespielt werden. Deshalb muß die genaue Länge oder die Bezugsperiode für die Viertelnote eindeutig definiert werden.
Dies kann man über einen Ausdruck im **SOUND**-Kommando erreichen, der als <Dauer> bezeichnet wird, einem ganzzahligen Wert mit der Einheit = 1/100 Sekunde (d. h. 100 = 1 Sek.)
Beachten Sie, daß die <Dauer> auch durch die Länge der <Lautstärken-Hüllkurve> festgelegt werden kann. Bedenken Sie dies, wenn Sie die **ENV**-Anweisung benutzen.

6.5 Andere Arten von Tönen

Mit dem Computer können auch Geräusche ('Weißes Rauschen') erzeugt werden, die nichts mit herkömmlichen Noten zu tun haben. Solche Effekte können z. B. als Hintergrund genützt werden um Melodien zu variieren, oder ganz eigenständig um Spezialeffekte zu erzielen. Eine Explosion ist so ein Geräusch, das durch die <Lautstärken Hüllkurve> geformt wird. Dabei wird die Frequenz – durch einen Zufallsgenerator gesteuert – innerhalb einer <Geräuschperiode> verändert, die im **SOUND**-Kommando festgelegt wurde. Dabei ist zu beachten, daß zwar 3 Kanäle zur Verfügung stehen und damit bis zu 3 <Tonperioden>, es kann aber nur eine <Geräuschperiode> festgelegt werden, die dann über die gewünschte Kanalkombination wiedergegeben wird.

Rendezvous für die Ton-Kanäle



6.6 Mehrfachklang und Kanäle

Die meisten Musikstücke sind für mindestens zwei Notenschlüssel, Ober- und Unterstimme geschrieben. Um dies auf dem CPC 464 nachvollziehen zu können, wurden 3 SOUND Kanäle – A, B und C – geschaffen. Über alle 3 kann unabhängig voneinander gespielt werden, natürlich aber auch zeitlich aufeinander abgestimmt. Die Kanalauswahl wird im SOUND-Kommando mit dem Parameter <Kanal-Status> getroffen.

6.7 Kanal-„Queues“

Für jeden SOUND-Kanal gibt es eine Queue (Warteschlange) mit einer Reihe von Tönen, die abgespielt werden. In jeder Queue ist Platz für fünf separate SOUND-Kommandos: ein aktives und vier, die warten. Das Betriebssystem des CPC464 ist in der Lage, andere Aufgaben zu erledigen, während die Tonwarteschlange abgespielt wird.

6.8 Kanal-Status

Das Schlüsselwort SQ wird verwendet um den Status eines Kanals, den Sie abfragen wollen, zu bestimmen. Es werden Informationen über freie Plätze ausgegeben. ON SQ GOSUB ist ein Unterbrechungsbefehl, der verwendet werden kann, um in den Programmteil, in dem die Musik erzeugt wird, zurückzuspringen.

6.9 Rendezvous und Haltestellen

Um die Kanäle zu synchronisieren steht eine sog. 'RENDEZVOUS'-Technik zur Verfügung. Dazu werden in zwei oder mehr Kanälen Markierungen gesetzt, die das gleichzeitige Einsetzen der entsprechenden SOUND-Kommandos erzwingen. Das ist eine sehr nützliche Einrichtung wenn z. B. Tonfolgen auf einem Kanal unterbrochen werden und auf dem anderen nicht.

Haltepunkte HOLD (siehe SOUND <Kanal-Status>) sind dort wichtig, wo generelle Kanalsynchronisation nötig ist (z. B. am Anfang einer Melodie), möglicherweise mit einer Verzögerung. Dann wird eine HOLD und RELEASE Folge benutzt.

6.10 Kommandofolge bei der Ton-Erzeugung

Das Format des Kommandos ist: `SOUND G,H,I,J,K,L,M`

mit

`G:` <Kanal-Status>
`H:` <Ton-Periode>
`I:` <Dauer>
`J:` <Lautstärke>
`K:` <Lautstärken-Hüllkurve>
`L:` <Ton-Hüllkurve>
`M:` <Geräusch-Periode>

In der `SOUND`-Anweisung sind alle Parameter Ganzzahlen. Nur die ersten beiden müssen unbedingt ausgegeben werden. Wenn die restlichen nicht angegeben werden, gelten die Standardwerte für diese Parameter. Diese werden bei der Beschreibung der Parameter erklärt.

6.10.1 Parameter Beschreibung

G: Kanal Status

Wertebereich 1 ... 255

Standardwert, wenn nicht belegt: keiner (Eingabe zwingend).

Mit dem CPC464 können gleichzeitig 3 verschiedene Töne gespielt werden, und zwar über die 3 vorhandenen `SOUND` Kanäle A, B und C. Die Eingabe erfolgt als dezimale Ganzzahl, aber wenn die Dezimalzahl in eine 8 Bit Darstellung umgewandelt wird, aktivieren die gesetzten Bits folgende Kommandos:

DEZIMAL	BIT	KOMMANDO
1	0 LSB	sende SOUND nach Kanal A
2	1	sende SOUND nach Kanal B
4	2	sende SOUND nach Kanal C
8	3	Rendezvous mit Kanal A
16	4	Rendezvous mit Kanal B
32	5	Rendezvous mit Kanal C
64	6	hold
128	7 MSB	flush

LSB = niederwertigstes Bit (Least Significant Bit)
MSB = höherwertigstes Bit (Most Significant Bit)

Beispiele für ganzzahlige Eingabe:

2 = sende den folgenden `SOUND` nach Kanal B
5 = sende den folgenden `SOUND` nach Kanal A und C
98 = 64+32+2
= sende den folgenden `SOUND` nach Kanal B, Rendezvous mit Kanal C und halte.

Wichtig: Wenn ein Rendezvous auf zwei oder mehreren Kanälen arrangiert werden soll, muß der <Kanal Status> eines jeden Kanals zur richtigen Zeit markiert werden.

Beim Setzen eines Haltepunktes stoppt der Ton und die Tonschlange dahinter wird solange eingefroren, bis ein **RELEASE** Kommando gegeben wird, oder sie mit einem späteren **SOUND** Kommando geleert wird (flush).

Wenn das Flush-Bit für einen Kanal gesetzt ist, wird der **SOUND**, der dabei mitgegeben wird, sofort ausgeführt. Die Tonschlange (Queue) bleibt dabei leer und der Kanalkopf inaktiv. Jegliche Töne, die zu diesem Zeitpunkt gerade gespielt werden, werden abgebrochen.

H: Ton Periode

Wertebereich 0 ... 4095

Standardwert, wenn nicht belegt: keiner (Eingabe zwingend).

Legt eine Periode und damit die Frequenz des Tons, der gespielt werden soll, fest. Die Frequenz bekommt man über folgende Formel: Frequenz = 125000/Periode.

Wenn Sie 0 angeben, wird keine Frequenz gesetzt. Das ist dann sinnvoll, wenn lediglich Geräusche erzeugt werden sollen.

I: Dauer

Jeder ganzzahlige Wert im Bereich −32768 ... +32767 ist erlaubt.

Standardwert, wenn nicht belegt: 20.

Bei Werten größer als 0 bedeutet der eingegebene Wert die Dauer in 1/100stel Sekunden. Wenn der Wert 0 ist, wird die Dauer durch die Hüllkurve der Lautstärke gesteuert.

Wenn die Zahl negativ ist, wird die Lautstärken-Hüllkurve dem positiven Wert der Zahl entsprechend oft wiederholt.

J: Lautstärke

Ein ganzzahliger Wert im Bereich 0 ... 15 (0 ... 7, wenn für die Lautstärke keine Hüllkurve angegeben ist).

Standardwert, wenn nicht belegt: 12 (4 ohne Lautstärken-Hüllkurve).

Das ist die Lautstärken am Anfang des Tons, die durch die Lautstärken-Hüllkurve verändert werden kann. 0 = keine Lautstärke.

K: Lautstärken-Hüllkurve

Ein ganzzahliger Wert im Bereich 0 ... 15.

Standardwert, wenn nicht belegt: 0.

Der Wert spezifiziert eine vorgegebene Hüllkurve, die mit **ENV** definiert wird. Eine feste Definition besteht für Lautstärken-Hüllkurve 0. Diese kann nicht geändert werden und ist auf 2 Sekunden auf der <Lautstärke Skala> festgelegt.

L: Ton-Hüllkurve

Ein ganzzahliger Wert im Bereich 0 ... 15.

Standardwert, wenn nicht belegt: 0.

Der Wert spezifiziert eine vorgegebene Hüllkurve, die mit `ENT` definiert wird. Eine feste Definition besteht für Ton-Hüllkurve 0, die nicht geändert werden kann und auf die ‚Ton Periode' festgesetzt ist.

M: Geräusch-Periode

Ein ganzzahliger Wert im Bereich 0 ... 15.

Standardwert, wenn nicht belegt: 0.

Legt das Geräusch (noise) fest, das dem `SOUND` hinzugefügt wird. Wenn nichts oder der Wert 0 angegeben wird, dann wird kein Geräusch hinzugefügt. Es kann immer nur eine Geräusch-Periode definiert werden. Das heißt, daß alle Kanäle mit dem gleichen Geräusch versorgt werden.

6.11 Das `ENV` Kommando und Hüllkurven für die Lautstärke

Anschwellen und Abklingen der Lautstärke geben einem Ton erst Leben. Mit diesem Kommando (`ENV` = `EN`velope `V`olume) kann man die Gestalt einer Note formen. Zeichnen Sie zuerst einmal die Gestalt der Note auf Papier, indem Sie die numerische Befehlsfolge umsetzen. Dazu folgendes Beispiel:

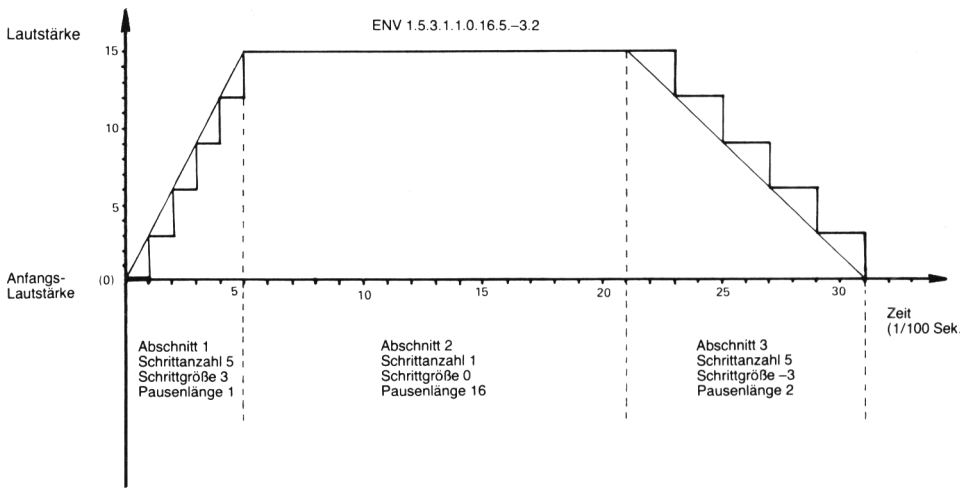


Bild 3: Beispiel für die Lautstärken-Hüllkurve

Die Gestalt muß in Zahlen umgesetzt werden, um sie in einem Befehl angeben zu können. Unterteilen Sie die Gestalt in bis zu 5 Abschnitte. In jedem Abschnitt hat die Gestalt eine gerade Linie. Unterteilen Sie jetzt jeden Abschnitt in Schritte, wobei die Anzahl der gewählten Schritte als <Schrittzahl> bezeichnet wird und die Länge eines Schrittes die <Pausenlänge> ist (1 = 1/100stel Sekunde).

Die Schritte nehmen um einen bestimmten Betrag, die <Schrittgröße>, zu oder ab. Wenn in einem Abschnitt keine Schritte angegeben sind, wird die Lautstärke nicht verändert und an den nächsten Abschnitt weitergegeben.

Beachten Sie: Wenn man versucht ein Musikinstrument zu simulieren, so muß man oft bei jeder Note am Anfang und am Ende die Lautstärke auf Null setzen. Deshalb sollte dann die Anfangslautstärke im **SOUND** Kommando auf Null gesetzt und jegliche Lautstärke nur über die Hüllkurve zugegeben werden.

Format des Kommandos:

ENV N,P1,Q1,R1,P2,Q2,R2,P3,Q3,R3,P4,Q4,R4,P5,Q5,R5

N: Nummer der Hüllkurve	Wertebereich 1 ... 15
P1 ... 5: Schrittzahl (Abschnitte 1 ... 5)	Wertebereich 0 ... 127
Q1 ... 5: Schrittgröße (Abschnitte 1 ... 5)	Wertebereich −128 ... +127
R1 ... 5: Pausenlänge (Abschnitte 1 ... 5)	Wertebereich 0 ... 125

Die Nummer der Hüllkurve muß angegeben werden. Ebenso muß ein kompletter Abschnitt angegeben werden, die Anzahl der Abschnitte ist beliebig. Wenn Sie z. B. die Abschnitte 4 und 5 weglassen, dann gibt es nur 3 Abschnitte.

6.12 Das ENT Kommando und Hüllkurven für den Ton

Diese sind genauso aufgebaut wie Hüllkurven für die Lautstärke, nur daß hier kleine Frequenzschwankungen einer Note erzeugt werden, so wie beim Vibrato. Wenn Sie sich für den Frequenzverlauf der Note entschieden haben, skizzieren Sie ihn und teilen Sie ihn in Abschnitte und Schritte ein. Dazu ein Beispiel:

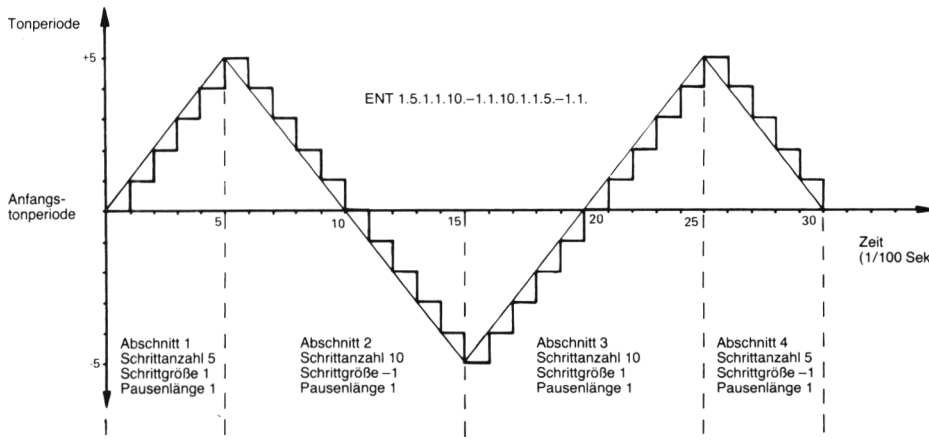


Bild 4: Kennlinie einer Ton-Hüllkurve

Der wichtigste Unterschied zwischen Hüllkurven für Lautstärke und Hüllkurven für Ton besteht darin, daß Ton-Hüllkurven nicht die Dauer einer Note beeinflussen, die vorher in einem **SOUND** Kommando oder einer Lautstärken-Hüllkurve definiert wurde. Wenn die Ton-Hüllkurve vor der Note zu Ende ist, bleibt die letzte Tonhöhe für den verbleibenden Rest der Zeit erhalten. Sollte sie jedoch länger sein, als die Zeitskala der Note, so entfallen die verbleibenden Abschnitte.

Um eine Ton-Hüllkurve während der Dauer des Klangs zu wiederholen, verwenden Sie eine negative Nummer für die Hüllkurve. (Beachten Sie, daß sie im **SOUND** Kommando nicht negativ aufgerufen wird.)

Format des Kommandos:

`ENT` S,T1,V1,W1,T2,V2,W2,T3,V3,W3,T4,V4,W4,T5,V5,W5

S: Hüllkurven-Nummer	Wertebereich 1 ... 15 (– für Wiederhl.)
T1 ... 5: Schrittzahl (Abschnitt 1 ... 5)	Wertebereich 0 ... 239
V1 ... 5: Schrittgröße (Abschnitt 1 ... 5)	Wertebereich −128 ... +127
W1 ... 5: Pausenlänge (Abschnitt 1 ... 5)	Wertebereich 0 ... 255 (1/100stel Sekunde)

S muß angegeben werden. Jeder Abschnitt, der angegeben ist, muß vollständig sein.

Wenn eine <Hüllkurven-Nummer> definiert wird, werden alle früheren zurückgesetzt. Wenn man eine Hüllkurve ohne Abschnitte angibt, werden alle Werte für diese Hüllkurve auf Null gesetzt.

6.13 Weitere zugehörige Funktionen und Kommandos

`SQ(x)`

x ist die Kanal-Nummer, eine bit-signifikante Nummer: 1, 2 oder 4.

Diese Werte stellen die Kanäle A, B, C dar, wie in der Bit-Tabelle für den Kanal-Status im `SOUND` Kommando gezeigt. Damit wird die Statusanzeige des Kanals abgefragt.

Die Funktion gibt als Antwort eine Ganzzahl im Bereich 0 bis 255. Die folgenden Bits sind für die Auswertung der Information von Bedeutung:

Bit 0 ... 2	zahl der freien Plätze in der getesteten Queue (Werte 0 ... 4)
Bit 3	Rendezvous mit Kanal A
Bit 4	Rendezvous mit Kanal B
Bit 5	Rendezvous mit Kanal C
Bit 6	haltepunkt am Queue-Anfang
Bit 7	Kanal gerade im Betrieb,

Bits 3 – 6 (Rendezvous und Halt) wurden bereits beim Kanal-Status beschrieben. Man hat bei diesem Test noch die Möglichkeit, das `ON SQ GOSUB` Kommando anzuschalten, das im nächsten Paragraph beschrieben wird.

`ON SQ GOSUB`

`ON SQ(y) GOSUB` <Zeilen-Nummer>

y ist die Kanalnummer: 1, 2 oder 4.

Dies ist ein Unterbrechungsbefehl, der erkennt, wenn eine Lücke in der Schlange für den angegebenen Kanal vorhanden ist. Er unterbricht dann die `SOUND` Unterroutine. Das `SQ` und das `SOUND` Kommando heben beide diese Unterbrechung auf, und die Unterroutine endet ganz normal mit `RETURN`. Alle Kanäle haben die gleiche Priorität und wenn eine Lücke im angegebenen Kanal gefunden wird, wird eine Unterbrechung erzeugt und die Routine stoppt sich selbst. Deswegen muß die Unterroutine die Unterbrechung aufheben, wenn Sie wieder verwendet werden soll.

RELEASE

RELEASE z

z bezeichnet die Kanalnummer (n) und ist eine Ganzzahl von 1 ... 7.

Wie im SOUND Kommando beschrieben, können die Kanäle mit Haltepunkten versehen werden. Mit RELEASE werden diese Punkte wieder aufgehoben. Die Eingabe der Kanäle ist bit-signifikant und gibt die Kombination der Kanäle wieder. RELEASE auf einem Kanal ohne Haltepunkte hat keine Auswirkung.

Bit 0 : Kanal A
Bit 1 : Kanal B
Bit 2 : Kanal C

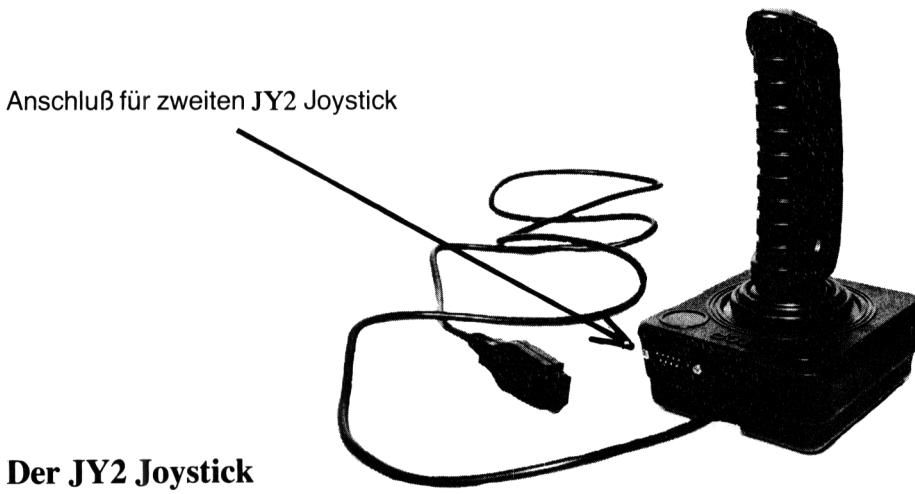
7 Drucker und Joysticks

Der CPC464 ist fertig ausgerüstet für den Betrieb mit einem oder zwei Joysticks und einem Parallel-Drucker mit Centronics-Schnittstelle.

Themen dieses Kapitels:

* Joysticks
* Parallel-Drucker
* Schnittstellen

Den **Amstrad** Joystick **JY2** sollten Sie dazukaufen, wenn Sie auf dem CPC464 Computerspiele mit Joystick-Bedienung verwenden und im Spiel auch ‚schießen' wollen. Der **JY2** wird an der 9poligen Buchse an der Linken Seite des Computers angeschlossen, über der **USER PORTS(I/O)** steht. Ein zweiter Joystick kann an einer entsprechenden Buchse im Sockel des ersten Joysticks angeschlossen werden. Eine Beschreibung aller Anschlüsse finden Sie im Anhang V am Ende dieses Handbuches.



Der JY2 Joystick

7.1 Joysticks

Die im **Amstrad** CPC464 eingebaute Software unterstützt einen oder zwei Joysticks, die wie ein Teil der Tastatur behandelt werden und deshalb mit `INKEY$` und `INKEY` genauso wie normale Tasten abgefragt werden können. Wenn nur ein Feuerknopf am Joystick vorhanden ist, wird er in der CPC464 Terminologie als ‚FIRE 2' betrachtet.

Es gibt eine spezielle Funktion, um die Joysticks direkt abzuprüfen. Es ist `JOY(0)` für den ersten und `JOY(1)` für den zweiten Joystick. Die Funktion zeigt einen Wert an, der die Bewegung des Joysticks zum Zeitpunkt der letzten Tastatur-Abprüfung wiedergibt. Da die Tastatur 50 Mal in der Sekunde abgeprüft wird, können Sie davon ausgehen, daß Sie immer den letzten Status der Joystick-Schalter erhalten.

Die Werte in der Spalte TASTE entsprechen den Werten der `INKEY` Funktion. Die Werte, die Sie unter `ABBILD` finden, entsprechen der Taste, die gedrückt werden kann, wenn kein zweiter Joystick vorhanden ist.

1. Joystick	JOY(0)	TASTE	2. Joystick	JOY(1)	TASTE	ABBILD
Auf	Bit 0	72	Auf	Bit 0	48	6
Ab	Bit 1	73	Ab	Bit 1	49	5
Links	Bit 2	74	Links	Bit 2	50	R
Rechts	Bit 3	75	Rechts	Bit 3	51	T
Feuer 2	Bit 4	76	Feuer 2	Bit 4	52	G
Feuer 1	Bit 5	77	Feuer 1	Bit 5	53	F

Wenn der zweite Joystick abgeprüft wird, kann der CPC464 nicht unterscheiden, ob der Joystick oder die Tastatur bedient wurde. In der Praxis kommt es wahrscheinlich nie zu solch einer Konstellation. Sie können also anstelle eines zweiten Joysticks auch die Tastatur verwenden.

Wenn Sie den **Amstrad JY2** Joystick verwenden, so ist der zweite Joystick der gleiche wie der erste. Er wird in eine Buchse auf der Seite des ersten Joysticks gesteckt. Man braucht kein spezielles Kabel, um den zweiten Joystick anzuschließen.

An der 9poligen Buchse, über der `USER PORTS(I/O)` steht, können Standard Joysticks angeschlossen werden, wie sie auch bei anderen Computern verwendet werden. Allerdings haben diese keinen Anschluß für einen zweiten Joystick, es sei denn, man benutzt einen speziellen Adapter. Sie sollten jedoch so einen Joystick nicht als zweiten Joystick seitlich in den **Amstrad JY2** einstecken.

Wenn Sie Software schreiben, sollten Sie am Anfang eines Programmes dem Benutzer die Möglichkeit geben, entweder den Joystick oder die Cursortasten zur Programmsteuerung zu verwenden (die `[COPY]` Taste oder eine andere anzugebende Taste könnte als Feuerknopf verwendet werden).

7.2 Drucker Schnittstelle

An den CPC464 kann ein Parallel Drucker mit der „Industriestandard" Schnittstelle CENTRONICS angeschlossen werden.

Das Kabel zwischen Drucker und Computer ist ganz einfach eine „eins zu eins" Verbindung. Das heißt, daß jeder Stift des einen Steckers mit dem entsprechenden des anderen verbunden ist. Auf der Platine des Computers sind zwei Stifte weniger als am Druckeranschluß, damit genormte Platinenstecker verwendet werden können.

Die Schnittstellenbeschreibung finden Sie im Anhang V.

Das Kabel sollte also Stift 1 des Computers mit Stift 1 des Druckers, 2 mit 2 usw. verbinden. Stift 18 und 36 des Druckers werden nicht mit dem Computer verbunden. Die untere Stiftreihe am Computer ist von 19 an aufwärts numeriert und nicht ab 18, wie man vielleicht erwarten würde. Die obere Reihe endet bei 17. Damit ist gewährleistet, daß jeder Draht zum jeweils gleich numerierten Stift auf der Computer-Seite und auf der Drucker-Seite geht.

Das *BUSY* Signal (Stift 11) wird verwendet um zu prüfen, ob der Drucker bereit ist. Der Computer wartet, wenn der Drucker *OFF LINE* ist.

Sie brauchen keine besonderen Befehle zugeben, außer im Programm die Druckausgabe auf die Gerätenummer (stream) 8 zu legen:

```
LIST #8
```

Dies bewirkt, daß das BASIC-Programm im Speicher am angeschlossenen Drucker gelistet wird, wenn es sich nicht um ein geschütztes Programm handelt.

Innerhalb eines Programmes können Sie den Drucker ganz einfach ansprechen:

```
PRINT #8, "Dies wird zum Drucker geschickt"
```

Viele Drucker brechen automatisch die Zeile um, wenn das Ende der Zeile erreicht wird. Schauen Sie dazu im Druckerhandbuch nach. Amstrad BASIC bricht ebenfalls die Zeile um, und zwar nach einer Länge, die im `WIDTH` Kommando angegeben ist. Der Standardwert für die Druckerbreite ist 132 und kann auf einen gewünschten neuen Wert gesetzt werden, z. B. `WIDTH 80`.

Wenn Sie die spezielle Zeilenbreite 255 wählen, dann überläßt Amstrad BASIC dem Drucker den Zeilenumbruch. Im BASIC gibt es einen Zähler für die Druckposition, der mit der `POS` Funktion abgefragt werden kann.

```
IF POS (#8) > 50 THEN GOTO 100
```

Der CPC464 erzeugt am Ende einer Zeile einen Zeilenvorschub `CHR$(10)` und einen Wagenrücklauf `CHR$(13)`. Der Drucker hat normalerweise einen Schalter, mit dem die richtige Form der Eingabe gewählt werden kann. Wenn sie drucken wird sofort klar, welches die richtige Schalterstellung ist.

7.3 Drucken von Graphik

Das Druckerhandbuch beschreibt die Steuerzeichen, die normalerweise diese Form haben:

```
PRINT CHR$(n)
```

Manche Drucker können sicherlich Zeichen ausgeben, die Ähnlichkeit haben mit vielen der Amstrad Graphik-Zeichen, wie sie im Anhang III zu finden sind. Aber es ist unwahrscheinlich, daß alle Zeichennummern übereinstimmen. Sie müssen dann eine eigene Tabelle mit den entsprechenden Zeichen aufbauen, die mit Ihrem Drucker übereinstimmen.

Obwohl die Drucker-Schnittstelle so ausgelegt ist, daß preisgünstige Matrixdrucker angeschlossen werden können, unterstützt sie auch Typenraddrucker mit entsprechender Schnittstelle, oder auch Graphikplotter und Farb-Tintenstrahldrucker. Der Schlüssel zur Kompatibilität ist die genormte parallele Schnittstelle.

8 Ausführliche Beschreibung des Amstrad CPC464 BASIC

Alle BASIC-Befehle – detailliert beschrieben, mit Beispielen und Querverweisen auf verwandte Befehle.

In diesem Kapitel finden Sie

* die Beschreibung des Befehlsaufbaus
* Sonderzeichen und ihre Bedeutung
* alle **Amstrad** BASIC-Befehle in alphabethischer Reihenfolge

Dieses Kapitel enthält eine exakte Zusammenstellung aller Befehle des BASIC, die im *ROM* des CPC464 enthalten sind. Die zur Verfügung stehenden Befehle umfassen nicht nur den üblichen BASIC-Sprachumfang, sondern unterstützen auch die besonderen Hardwaremöglichkeiten des CPC464.

8.1 **Notation** – Erklärung der bei der Beschreibung der Befehle verwendeten Zeichen

Sonderzeichen

`& oder &H`	kennzeichnet die nachfolgende Konstante als hexadezimal (sedezimal)
`&X`	kennzeichnet die nachfolgende Konstante als binär (dual)
`:`	trennt Befehle, die in derselben Zeile stehen
`#`	kennzeichnet die nachfolgende Konstante als Ein-/Ausgabeeinheit (stream).

Datentypen

Texte können 0 bis 255 Zeichen lang sein und werden durch `<Textausdruck>` `(<string expression>)` dargestellt. Texte können mit einem `'+'` Zeichen aneinandergehängt werden. Dabei darf die Gesamtlänge natürlich 255 Zeichen nicht überschreiten.

Ein <Numerischer Ausdruck> (<numeric expression>) kann entweder ganzzahlig oder reell sein, d. h., die Zahlenwerte werden ohne bzw. mit Dezimalpunkt dargestellt. Ein <Ganzzahliger Ausdruck> (<Integer expression>) kann die Werte von −32768 bis +32767 annehmen. Ein <Reeller Ausdruck> (<Real expression>) kann im Bereich von −1.7E+38 bis +1.7E+38 liegen (E bedeutet 'mal 10 hoch'). Reelle Größen haben eine Genauigkeit von etwas über neun Stellen. Die kleinste ungleich 0 darstellbare Zahl ist 2.9E−39.

% als letztes Zeichen des Variablennamens definiert die Variable als ganzzahlig, ! als reell und $ als Textgröße.

Ein <Numerischer Ausdruck> hat immer einen Zahlenwert als Ergebnis. Dabei kann es sich um eine echte Zahlenangabe, eine Variable (ganzzahlig oder reell) oder einen mathematischen Ausdruck handeln. Kurz, alles was keinen <Textausdruck> darstellt, ist erlaubt.

Die <Ein-/Ausgabeeinheit> stellt einen <Numerischen Ausdruck> dar, der z. B. den Bildschirm, den Drucker oder das Cassettengerät identifiziert.

Ein 'unzulässiger Ausdruck' ('improper argument') ergibt sich entweder, wenn ein <numerischer Ausdruck> einen Wert annimmt, der außerhalb des zulässigen Zahlenbereichs liegt, oder wenn ein Kommandoparameter falsch eingegeben ist.

BEFEHLE

Bitte beachten Sie, daß alle **Amstrad** BASIC-Befehlsbeschreibungen folgendermaßen aufgebaut sind:

BEFEHL
Befehlsschreibweise/Funktion
Beispiel
Beschreibung
Verwandte BEFEHLE

WICHTIG:

Befehle sind entweder
Kommandos, die sofort ausgeführt werden oder
Funktionen, die einen Wert liefern innerhalb eines Ausdrucks

Klammern
( ) sind Teile eines Kommandos oder einer Funktion. Anders dargestellte Klammern innerhalb der Befehlsschreibweise haben nur die Bedeutung einer Beschreibung und dürfen nicht mit eingegeben werden
[ ] umschließen wahlweise Ausdrücke
< > umschließen verschiedenste Ausdrücke, die in der Befehlsbeschreibung erklärt werden.

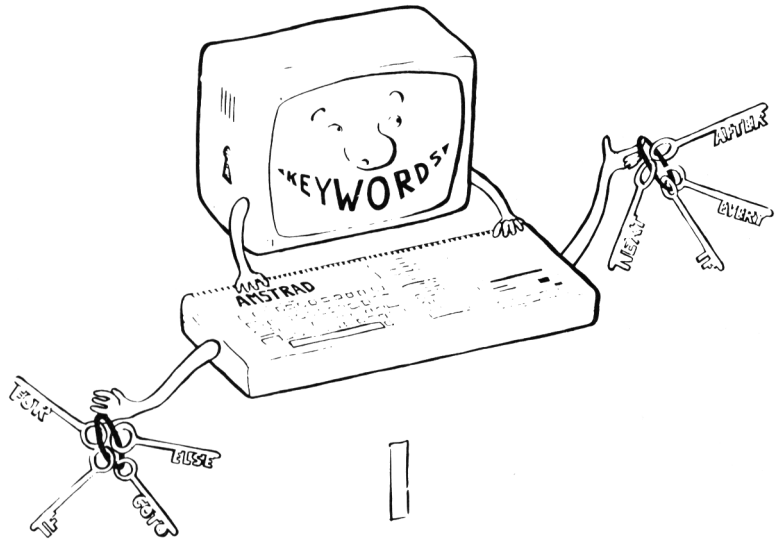
Anführungszeichen

Nur " "(doppeltes Anführungszeichen) sind Bestandteil der BASIC-Sprache. ' (einfache Anführungszeichen) dienen nur zur besonderen Hervorhebung innerhalb der Befehlsbeschreibung. ' dürfen nirgens im Befehl erscheinen, es sei denn, innerhalb eines <Textausdrucks>.

Eingaben

BASIC wandelt alle mit kleinen Buchstaben eingegebenen Befehle in Großbuchstaben um, wenn das Programm aufgelistet wird. Die verwendeten Beispiele sind alle mit Großbuchstaben beschrieben, wie sie bei der Auflistung erscheinen. Wenn Sie die Beispiele mit kleinen Buchstaben eingeben, können Sie Ihre Tippfehler leichter finden, da alle falschgeschriebenen Befehle bei der Auflistung weiterhin kleingeschrieben erscheinen.

Befehle müssen durch Trennzeichen begrenzt werden, da Amstrad BASIC erlaubt, Befehlswörter in Variablennamen zu verwenden: z. B. end2 und `LISTCODE` werden von Amstrad BASIC als Variablen angenommen.



ABS

```
ABS (<numerischer Ausdruck>)
```

```
PRINT ABS(-67.98)
 67.98
```

FUNKTION: Gibt den absoluten Wert des angegebenen Ausdrucks zurück, d. h., negative Zahlen werden positiv zurückgegeben.

AFTER

```
AFTER <ganzzahliger Ausdruck>[,<ganzzahliger Aus-
druck>] GOSUB <Zeilennummer>
AFTER 200,2 GOSUB 320
```

KOMMANDO: Ruft in bestimmten Zeitabständen ein Unterprogramm auf. Der erste <ganzzahlige Ausdruck> beschreibt das Zeitintervall in Einheiten zu 0,02 Sekunden. Der zweite <ganzzahlige Ausdruck> (im Bereich zwischen 0 bis 3) gibt die Uhr (4 mögliche) an, die verwendet werden soll.

Verwandte Befehle: `EVERY`, `REMAIN`

ASC

```
ASC (<Textausdruck>)
```

```
PRINT ASC("X")
 88
```

FUNKTION: Gibt den numerischen Wert nur des ersten Zeichens des Textausdruckes zurück, falls ASCII-Zeichen verwendet werden.

Verwandte Befehle: `CHR$`

ATN

```
ATN(<numerischer Ausdruck>)
```

```
PRINT ATN(1)
 0.785398163
```

FUNKTION: Gibt einen Wert zwischen −PI/2 und +PI/2 im Bogenmaß zurück, der sich aus dem Arcustangens des angegebenen `<numerischen Ausdrucks>` errechnet.

Verwandte Befehle: `SIN, COS, TAN, DEG, RAD`

AUTO

```
AUTO [<Zeilennummer>][,<Schrittweite>]
AUTO 100,50
```

KOMMANDO: Erzeugt automatisch Zeilennummern. `<Zeilennummer>` gibt die erste Zeilennummer an, mit der begonnen wird, wenn z. B. ein Programm verlängert werden soll. Die `<Schrittweite>` bestimmt die Differenz zur nächsten Zeilennummer. Wird einer der beiden Eingabewerte nicht angegeben, so wird 10 als Standardwert angenommen.

Falls eine Zeile mit derselben Zeilennummer schon vorhanden ist, warnt BASIC mit einem * nach der ausgegebenen Zeilennummer, um auf die Gefahr der Überschreibung hinzuweisen.

BIN$

```
BIN$ (<vorzeichenloser ganzzahliger Ausdruck> [,
<ganzzahliger Ausdruck>]
```

```
PRINT BIN$(64,8)
                    01Betriebsferien
                    30.7.=10.8.1984

01000000
```

FUNKTION: Gibt einen Text zurück, der den Wert des `<vorzeichenlosen ganzzahligen Ausdrucks>` in binärer (dualer) Form enthält. Falls der zweite `<ganzzahlige Ausdruck>` größer ist, als die Anzahl der sowieso benötigten Zeichen, wird der Text mit führenden Nullen versehen, bis zu der durch den zweiten `<ganzzahligen Ausdruck>` angegebenen Gesamtlänge.

Verwandte Befehle: `HEX$, STR$`

BORDER

```
BORDER <Farbe>[,<Farbe>]
```

```
BORDER 3,2
```

KOMMANDO: Die nicht beschreibbare Randfläche des Bildschirms wird in der angegebenen `<Farbe>` dargestellt. Wird eine zweite `<Farbe>` angegeben, so wechselt die Randfläche immer wieder ihr Aussehen zwischen den beiden Farben. Das Zeitintervall zwischen dem Farbwechsel wird durch das Kommando `SPEED INK` festgelegt. `<Farbe>` darf Werte von 0 bis 26 haben.

Verwandte Befehle: `SPEED INK`

CALL

```
CALL <Adresse>[,<Liste von Parametern>]
CALL &BD19
```

KOMMANDO: Springt in ein Unterprogramm außerhalb des BASIC-Bereiches, das an der angegebenen Speicheradresse beginnt. Bei der Verwendung dieses Kommandos ist äußerste Vorsicht geboten. Es sollte nur von wirklich erfahrenen Programmierern angewendet werden. Der als Beispiel verwendete `CALL` ist relativ harmlos, da er auf den nächsten Bildaufbau wartet (erst wenn der Elektronenstrahl das neue Bild ganz aufgebaut hat, wird es auch sichtbar). Damit können Bewegungen von Zeichen auf dem Bildschirm wirklichkeitsgetreu dargestellt werden.

Verwandte Befehle: `UNT`

CAT

```
CAT
```

```
CAT
```

KOMMANDO: Liest die Cassette und gibt die Namen aller gefundenen Dateien aus. Der Befehl hat keine Auswirkungen auf das sich augenblicklich im Speicher befindende Programm. Damit kann das Kommando dazu benützt werden, zu überprüfen, ob ein sichergestelltes Programm auch ordentlich auf der Cassette abgespeichert wurde, bevor man den Speicher für ein anderes Programm verwendet. Das Kommando verlangt, daß die PLAY-Taste des Cassettengerätes gedrückt wird und danach eine beliebige Taste gedrückt wird `("PRESS PLAY then any key:")` und gibt für jeden gefundenen Datenblock eine Meldung aus in der Form:

```
Dateiname Blocknummer Dateikennzeichen Ok
```

Die Dateikennzeichen bedeuten:

- `$` BASIC-Programm
- `%` geschütztes BASIC-Programm
- `*` ASCII-Textdatei
- `&` Binäre Datei in Maschinensprache

Es können auch andere Dateikennzeichen auftreten, falls die Datei nicht von BASIC erzeugt wurde.

Verwandte Befehle: `LOAD,RUN,SAVE`

CHAIN
CHAIN MERGE

```
CHAIN <Dateiname>[,<Zeilennummernausdruck>]
CHAIN MERGE <Dateiname>[,<Zeilennummernausdruck>]
[,DELETE<Zeilenbereich>]
```

```
CHAIN "TEST", 350
```

KOMMANDO: `CHAIN` lädt das Programm `<Dateiname>` von der Cassette in den Speicher, wobei das augenblicklich geladene Programm gelöscht wird. `CHAIN MERGE` fügt das Programm `<Dateiname>` von der Cassette in das im Speicher befindliche Programm entsprechend der enthaltenen Zeilennummern ein. Der `<Zeilennummernausdruck>` gibt an, in welcher Zeile das Programm nach dem Laden beginnen soll. Fehlt die Startzeilenangabe, beginnt BASIC bei der niedrigsten Zeilennummer. Mit `DELETE <Zeilenbereich>` wird vor dem Laden der angegebene Bereich gelöscht.

Wird kein `<Dateiname>` angegeben, so lädt BASIC das erste gefundene Programm. Falls vor dem `<Dateinamen>` ein `!` eingegeben wird, so werden die üblichen Meldungen des Cassettenlesens unterdrückt.

`CHAIN MERGE` erhält alle bisher benutzten Variablen, aber selbstdefinierte Funktionen und offene Dateien gehen verloren. Ein gesetzter `ON ERROR GOTO`-Sprung wird ausgeschaltet, ein `RESTORE` wird ausgeführt und die mit `DEFINT,DEFREAL` und `DEFSTR` gesetzten Variablentypdefinitionen werden zurückgesetzt. Außerdem werden alle aktiven `FOR-, WHILE-` und `GOSUB-`Kommandos vergessen. Geschützte BASIC-Programme können nicht eingefügt werden.

Verwandte Befehle: `LOAD, MERGE`

CHR$

```
CHR$ (<numerischer Ausdruck>)
```

```
PRINT CHR$(100)
 d
```

FUNKTION: Gibt das Textzeichen zurück, auf das der `<ganzzahlige Ausdruck>` in der Amstrad CPC464 Zeichentabelle (siehe Anhang III) zeigt.

Verwandte Befehle: `ASC,LEFT$,RIGHT$,MID$,STR$`

CINT

```
CINT (<numerischer Ausdruck>)
```

```
10 n=578.76543
20 PRINT CINT(n)
RUN
 579
```

FUNKTION: Gibt den gerundeten ganzzahligen Wert des `<numerischen Ausdrucks>` zurück im Wertbereich von –32768 bis +32767

Verwandte Befehle: `CREAL,INT,FIX,ROUND,UNT`

CLEAR

CLEAR

CLEAR

KOMMANDO: Setzt alle Variablen auf Null und vergißt alle Dateien.

CLG

CLG[<Farbstift>]

CLG

KOMMANDO: Füllt den Graphikschirm mit der dem <Farbstift> (INK) zugeordneten Farbe. Ist kein <Farbstift> angegeben, wird der beim letzten CLG-Kommando angegebene Farbstift verwendet, ansonsten der Farbstift 0.

Verwandte Befehle: CLS, ORIGIN

CLOSEIN

CLOSEIN

CLOSEIN

KOMMANDO: Schließt die Eingabe einer Datei vom Cassettengerät. Kommandos wie NEW und CHAIN MERGE vergessen jede offene Datei.

Verwandte Befehle: OPENIN, CLOSEOUT

CLOSEOUT

CLOSEOUT

CLOSEOUT

KOMMANDO: Schließt die Ausgabe einer Datei auf das Cassettengerät.

Verwandte Befehle: OPENOUT, CLOSEIN

CLS

CLS[#<Ein-Ausgabeeinheit>]

CLS

KOMMANDO: Füllt den mit #<Ein-Ausgabeeinheit> angegebenen Bildschirmbereich mit der für den entsprechenden Hintergrund festgelegten Farbe.

CONT

CONT

CONT

KOMMANDO: Setzt ein durch *Break*, STOP oder END abgebrochenes Programm fort, falls es nicht geändert wurde. Kommandos dürfen zwischen Abbruch und Fortsetzung eingegeben werden.

COS

COS(<numerischer Ausdruck>)

```
?COS(34)
-0.848570274
```

und

```
deg:?cos(34)
 0.829037573
```

FUNKTION: Gibt den Wert des Cosinus des <numerischen Ausdrucks> zurück. Die Funktion versteht den <numerischen Ausdruck> als im Bogenmaß angegeben, solange nicht durch das DEG-Kommando auf Winkelgradmaß umgeschaltet wurde. Beachten Sie im Beispiel die Verwendung des ? als Kurzform für das PRINT-Kommando und ebenso die Verwendung von Kleinbuchstaben für Befehle – eine Möglichkeit, die vom **Amstrad** BASIC voll verstanden wird.

Verwandte Befehle: SIN,TAN,ATN,DEG,RAD

CREAL

CREAL(<numerischer Ausdruck>)

```
5 DEFINT n
10 n=75.765
20 d=n/34.6
30 PRINT d
40 PRINT CREAL(n)
50 PRINT n/55.4
run
2.19653179
76
1.37184116
Ready
```

FUNKTION: Gibt den <numerischen Ausdruck> in reeller Form zurück, d. h. die Zahl wird in eine mit Dezimalpunkt umgewandelt.

Verwandte Befehle: CINT

DATA

DATA <Liste von:<Konstanten>>

```
10 REM Namensliste
20 DIM Vorname$(3)
30 DIM Nachname$(3)
40 FOR n=1 to 3
50 READ Vorname$(n)
60 READ Nachname$(n)
65 PRINT Vorname$(n);" "Nachname$(n)
70 DATA Bernhard, Peter, Wulf, Koch, Meier, Huber
90 NEXT
```

KOMMANDO: Stellt Datenwerte dem Programm zur Verfügung. Es ist eine der meistgenützten Möglichkeiten in BASIC, Datenwerte in DATA-Kommandos zusammenzustellen, um sie bei Bedarf abzurufen. Die angegebenen Daten müssen mit der Art der Variablen übereinstimmen, die sie aufnehmen sollen. DATA-Kommandos können überall im Programm stehen.

Verwandte Befehle: READ,RESTORE

DEF FN

DEF FN<Name>[(<Formalparameter>)]=<allgemeiner Ausdruck>

```
10 DEF FNzins(betrag)=1.14*betrag
20 INPUT "Wie hoch ist der Betrag?";betrag
30 PRINT "Die Summe aus Zinsen und Betrag nach
   einem Jahr ist"; FNzins(betrag)
```

KOMMANDO: BASIC ermöglicht es, innerhalb eines Programms sog. Funktionen zu definieren, die einen Wert zurückgeben. Die mit DEF FN-Kommandos erzeugten Funktionen gelten innerhalb des Programms und arbeiten auf dieselbe Art und Weise wie die in BASIC eingebauten Funktionen (z. B. COS).

Die Funktion kann überall im Programm verwendet werden. Die zu übergebenden Variablen müssen im Typ mit den in der Funktion verwendeten übereinstimmen. Außerdem sollte die Definition der Funktion im Programmablauf nur einmal durchlaufen werden.

DEFINT
DEFSTR
DEFREAL

DEFtyp<Buchstabenbereich(e)>

```
DEFINT I-N
DEFSTR A,W-Z
DEFREAL
```

KOMMANDO: Legt fest, daß alle Variablen, die mit den angegebenen <Buchstaben> beginnen, vom Variablen-typ <typ> sind, wobei INT ganzzahlig (Integer), STR Text (string) und REAL reell (mit Dezimalpunkt) (REAL) bedeutet. Dabei werden Groß- und Kleinbuchstaben nicht unterschieden.

Verwandte Befehle: LOAD,RUN,CHAIN,NEW,CLEAR

DEG

DEG

```
DEG
```

KOMMANDO: Schaltet auf Winkelgradmaß um. Standardmäßig werden die in den Winkelfunktionen wie SIN und COS verwendeten Werte als im Bogenmaß angegeben verwendet. Das Kommando schaltet nun die Interpretation auf Winkelgradmaß um, bis sie durch CLEAR oder RAD oder das Laden eines neuen Programms wieder aufgehoben wird.

Verwandte Befehle: RAD

DELETE

DELETE <Zeilenbereich>

DELETE 100-200

KOMMANDO: Löscht alle Programmzeilen innerhalb des <Zeilenbereichs>. Da die gelöschten Zeilen nur wieder neu eingegeben werden können, sollte das Kommando mit Vorsicht benützt und auf Schreibfehler überprüft werden, bevor es abgeschickt wird.

Verwandte Befehle: NEW

DI

DI

```
10 CLS
20 TAG
30 EVERY 10 GOSUB 100
40 X1=RND*320:X2=RND*320
50 Y=200+RND*200
60 FOR X=320-X1 TO 320+X2 STEP 2
70 DI:PLOT 320,0:MOVE X-2,
   Y:PRINT " ";:MOVE X,Y:PRINT "#";:EI
80 NEXT
90 GOTO 40
100 MOVE 320,0
110 DRAW X+8,Y-16,0
120 RETURN
```

KOMMANDO: Verhindert alle Unterbrechungen außer durch *Break*, bis sie durch das EI-Kommando oder das RETURN-Kommando eines Unterbrechungs-GOSUB-Unterprogramms wieder ermöglicht werden. Das Kommando wird benutzt, um ein Programm ohne Unterbrechung Befehl für Befehl ablaufen zu lassen, falls z. B. zwei Programmteile ein und dasselbe Mittel benutzen wollen. In obigem Beispiel verwenden das Hauptprogramm und das Unterprogramm jeweils den graphischen Bildschirm.

Verwandte Befehle: EI

DIM

DIM Liste von:<indizierten Variablen>

```
10 CLS:PRINT "Eingabe von 5 Namen...":PRINT
20 DIM B$(5)
30 FOR N=1 TO 5
40 PRINT "Namen"N"bitte;
50 INPUT B$(N)
60 NEXT
70 FOR N=1 TO 5
80 PRINT "Wie klug von Ihnen";B$(N);
   "den CPC464 zu kaufen"
90 NEXT
```

KOMMANDO: Stellt Speicherbereich für Felder (Matrizen) bereit und legt den höchsten Tabellenwert (Index) fest. In BASIC muß die Indexobergrenze angegeben werden oder aber es wird der Standardwert 10 angenommen. Ist die Feldgröße einmal festgelegt, darf sie nicht geändert werden, da sonst ein Fehler auftritt. Eine <indizierte Variable> kann mit dem gleichen Variablennamen soviele Werte aufnehmen, wie durch die ganzzahligen Indexobergrenzen in der <Indexliste> angegeben sind. Man kann sich z. B. die erste Zahl in einer <Indexliste> vorstellen als die Stockwerke eines Parkhauses und die zweite dann als die Anzahl der Stellplätze je Stockwerk. Volles Verständnis für den Aufbau von Feldern ist eine wichtige Voraussetzung für fortgeschrittene BASIC-Programmierung. Die Größe eines Feldes ist einzig vom zur Verfügung stehenden Speicherbereich und der Vorstellungskraft des Programmierers begrenzt.

Verwandte Befehle: ERASE

DRAW

DRAW <x-Koordinate>,<y-Koordinate>[,<Farbstift>]

```
DRAW 200,200,13
```

KOMMANDO: Zeichnet eine Linie von der augenblicklichen Cursorposition zur angegebenen absoluten (x,y)-Position auf dem graphischen Bildschirm. Die Koordinatenangaben sind für alle drei Bildschirmmodi dieselben. Weitere Beispiele befinden sich in Kapitel 5.

Verwandte Befehle: DRAWR,PLOT,PLOTR,MOVE,MOVER,TEST,TESTR,
XPOS,YPOS,ORIGIN

DRAWR

DRAWR <x-Versatz>,y-Versatz>[,Farbstift>]

```
DRAWR 200,200,13
```

KOMMANDO: Zeichnet eine Linie von der augenblicklichen Cursorposition zur angegebenen relativen (x,y)-Position, also um x und y Zeichenpositionen versetzt.

Verwandte Befehle: DRAW,PLOT,PLOTR,MOVE,MOVER,TEST,TESTR,
XPOS,YPOS,ORIGIN

EDIT

EDIT <Zeilennummer>

```
EDIT 110
```

KOMMANDO: Bearbeiten einer bestimmten Programmzeile. Siehe Kapitel 1.

Verwandte Befehle: LIST

EI

EI

```
EI
```

KOMMANDO: Um Unterbrechungen nach einem DI-Kommando wieder zu ermöglichen.

Verwandte Befehle: DI

END

END

```
END
```

KOMMANDO: Beenden des Programms. Ein Programm endet auch mit der letzten Befehlszeile. END schließt alle Dateien und schaltet in den Direkteingabemodus um. Noch gespeicherte Tonfolgen werden bis zu ihrem Ende weitergespielt.

Verwandte Befehle: STOP

ENT

ENT <Folgenummer>[,<Veränderungsfolgen>]

```
10 ENT 1,100,2 20
20 SOUND 1,100,1000,4,0,1
```

KOMMANDO: Ermöglicht einen auszugebenden Ton in seiner Höhe zu variieren. Eine Ton<Veränderungsfolge> bestimmt, wie die Tonhöhe variiert werden soll. Eine ausführliche Beschreibung befindet sich in Kapitel 6 und Anhang VII.
Die <Folgenummer> ist ein <ganzzahliger Ausdruck>, dessen Wert zwischen 1 und 15 liegen muß, und kennzeichnet die Tonveränderungsfolge. Falls die <Folgenummer> negativ ist, wird die <Veränderungsfolge> wiederholt.

Bis zu fünf <Veränderungsfolgen> können jeweils angegeben werden, in der Form <Schrittanzahl>, <Pausezeichen>, oder: <Tonperiode>, <Pausezeichen>.

Die erste Form beschreibt eine schrittweise Veränderung relativ zur momentanen Tonperiode, die zweite entspricht einem absoluten Setzen der Tonperiode. Dabei bedeutet: <Schrittanzahl> einen <ganzzahligen Ausdruck> zwischen 0 und 239, der die Anzahl der auszuführenden Einzelveränderungen darstellt, <Schrittweite> einen <ganzzahligen Ausdruck> zwischen -128 und +127, der die Differenz zwischen den jeweiligen Tonperioden darstellt.

<Pausezeit> einen <ganzzahligen Ausdruck> zwischen 0 und 255, der die Wartezeit zwischen den jeweiligen Tonveränderungen darstellt. Dabei wird die Zeit in Einheiten zu 0.01 Sekunden gerechnet, außerdem wird für die Angabe 0 der Faktor 256 gerechnet.

<Tonperiode> einen <ganzzahligen Ausdruck> zwischen 0 und 4095, der den neuen Ton angibt.

Das `SOUND`-Kommando setzt die Start-Tonperiode und kann eine der 15 Folgenummern ansprechen. Wird keine oder eine nicht definierte Folgenummer angegeben, behält der angegebene Ton immer seine Originalhöhe.

Eine Tonveränderung hat keine Auswirkungen auf die Dauer des Tones. Sind noch Schritte in der Veränderungsfolge übrig, obwohl der Ton schon geendet hat, werden diese nicht mehr ausgeführt.

Eine zu wiederholende Tonveränderungsfolge wird immer wieder von vorne begonnen, so lange der Ton dauert.

Die im Kommando angegebenen Werte werden bei der Ausführung gespeichert und können später immer wieder abgerufen werden. Das Kommando muß also nicht wiederholt werden.

Jedesmal, wenn eine Tonveränderungsfolge mit der gleichen <Folgenummer> definiert wird, werden die alten Werte überschrieben. Findet eine solche Änderung statt, während ein `SOUND`-Kommando sie benutzt, so entstehen unbestimmte, aber vielleicht auch interessante Effekte.

Wird ein `ENT`-Kommando ohne <Veränderungsfolgen> angegeben, so werden die gespeicherten Werte der angesprochenen <Folgenummer> gelöscht.

Verwandte Befehle: `ENV`, `SOUND`

ENV

`ENV` <Folgenummer>[<Veränderungsfolgen>]

```
10 ENV 1,100,2,20
20 SOUND 1,100,1000,4,1
```

KOMMANDO: Ermöglicht einen auszugebenden Ton in seiner Lautstärke zu verändern und beschreibt, wie die Lautstärke verändert werden soll. Dabei kann die Schrittanzahl zwischen 0 und 127, die Schrittweite zwischen -128 und +127 und die Pausezeit in Einheiten zu 0.01 Sekunden zwischen 1 und 256 liegen.

Die <Folgenummer> ist ein <ganzzahliger Ausdruck>, dessen Wert zwischen 1 und 15 liegen muß und kennzeichnet die Lautstärkeveränderungsfolge.

Bis zu fünf <Veränderungsfolgen> können jeweils angegeben werden in der Form: <Schrittanzahl>, <Schrittweite>, <Pausezeit>

oder = <Registerwert>, <Veränderungsperiode>.

Die erste Form beschreibt eine Veränderung mit Hilfe der Software. Dabei bedeutet:

<Schrittanzahl> einen <ganzzahligen Ausdruck> zwischen 0 und 127, der die Anzahl der auszuführenden Einzelveränderungen darstellt. <Schrittweite> einen <ganzzahligen Ausdruck> zwischen -128 und +127, der die Differenz zwischen den jeweiligen Lautstärken (Amplituden) darstellt oder aber, falls die <Schrittanzahl> gleich 0 ist, den absoluten Lautstärkewert.

<Pausezeit> einen <ganzzahligen Ausdruck> zwischen 0 und 255, der die Wartezeit zwischen den jeweiligen Lautstärkeveränderungen darstellt. Dabei wird die Zeit in Einheiten zu 0.01 Sekunden gerechnet. Außerdem wird für die Angabe 0 der Faktor 256 gerechnet.

Die zweite Form beschreibt eine Veränderung durch direkten Zugriff auf den Tongenerator. Dabei bedeutet <Registerwert> den in das Lautstärkeregister <Register 15 oktal> zu setzenden Wert. <Veränderungsperiode> die in die Veränderungsfolgenregister (Register 13 und 14 oktal) zu setzenden Werte.

Setzt man direkt die Hardwareregister, so wird die Lautstärkeveränderung sofort ausgeführt. Es ist daher ratsam, vor der nächsten Lautstärkeveränderung eine Tonpause einzulegen. Gibt es keine weitere Lautstärkeveränderung, so wird eine Pause von 2 Sekunden ausgeführt.

Das **SOUND**-Kommando setzt die Start-Lautstärke und kann eine der 15 Folgenummern ansprechen. Wird keine oder eine nicht definierte Folgenummer angegeben, bleibt die lautstärke immer gleich. Wird zwar eine <Schrittanzahl>, aber eine <Schrittweite> 0 angegeben, so bleibt die Lautstärke gleich.

Die im Kommando angegebenen Werte werden bei der Ausführung gespeichert und können später immer wieder abgerufen werden. Das Kommando muß also nicht wiederholt werden.

Jedesmal, wenn eine Lautstärkeveränderungsfolge mit der gleichen <Folgenummer> definiert wird, werden die alten Werte überschrieben. Findet eine solche Änderung statt, während ein **SOUND**-Kommando sie benützt, so entstehen unbestimmte, aber vielleicht auch interessante Effekte. Wird ein **ENV**-Kommando ohne <Veränderungsfolgen> angegeben, so werden die gespeicherten Werte der angesprochenen <Folgenummer> gelöscht.

Verwandte Befehle: **ENT**, **SOUND**

EOF

EOF

```
PRINT EOF
-1
```

FUNKTION: Gibt den Wert -1 (wahr) zurück, wenn eine Eingabe von Cassette das Dateiende erreicht hat, ansonsten den Wert 0 (falsch).

Verwandte Befehle: OPENIN

ERASE

ERASE<Liste von <Variablennamen>>

```
ERASE A,B$
```

KOMMANDO: Löscht die angegebenen Variablen. Falls ein Feld nicht mehr benötigt wird, kann es gelöscht werden, um den Speicherplatz anderweitig verwenden zu können.

ERR
ERL

ERR
ERL

```
10 CLS
20 ON ERROR GOTO 1000
30 DATA SALLY,EMMA,JOANNE,HELEN,GEMMA
40 READ A$
50 PRINT A$
60 GOTO 40
70 REM hier beginnt die error-Behandlung
1000 IF ERR=4 THEN PRINT "Ich habe einen Fehler in
     einer DATA-Anweisung festgestellt!"
1005 PRINT "Es fehlen Wertangaben."
1010 IF ERL<70 AND ERL>20 THEN PRINT
     "Der Fehler trat in den Zeilen 30-60 auf."
1020 END
```

VARIABLEN: Diese Variablen werden in Fehlerbehandlungsroutinen verwendet, um festzustellen, welcher Fehler mit welcher Nummer in welcher Zeile aufgetreten ist. Fehlermeldungen sind in Anhang VIII beschrieben.

Verwandte Befehle: ON ERROR,ERROR

ERROR

ERROR <ganzzahliger Ausdruck>

```
ERROR 17
```

KOMMANDO: Führt eine Behandlung des Fehlers mit dieser Nummer aus. Die Fehlernummer kann eine sein, die BASIC schon verwendet (Anhang VIII). In diesem Falle wird die dafür vorgesehene Behandlung ausgeführt. Im anderen Fall wird der Fehler nur angezeigt und das Programm abgebrochen.

Verwandte Befehle: ON ERROR,ERR,ERL

EVERY

EVERY <Ganzzahliger Ausdruck>,[<Ganzzahliger Ausdruck>] GOSUB <Zeilennummer>

```
EVERY 500,2 GOSUB 50
```

KOMMANDO: Der CPC464 enthält eine Echtzeituhr. Das EVERY Kommando ermöglicht einem BASIC Programm in regelmäßigen Abständen Unterprogramme aufzurufen. Vier verschiedene Zeitsignalgeber stehen zur Verfügung, die durch die Werte 0 bis 3 mit dem zweiten <ganzzahligen Ausdruck> einem Unterprogramm zugeordnet werden können.

Verwandte Befehle: AFTER, REMAIN

EXP

EXP(<Numerischer Ausdruck>)

```
PRINT EXP(6.876)
 968.743625
```

FUNKTION: Gibt den Wert von 'e' potenziert mit dem <Numerischen Ausdruck> zurück. 'e' ist der natürliche Logarithmus von 1, also angenähert 2.7182818.

Verwandte Befehle: LOG

FIX

FIX(<Numerischer Ausdruck>)

```
PRINT FIX(9.99999)
 9
```

FUNKTION: Gibt die ganze Zahl zurück, die durch Abschneiden der Dezimalstellen entsteht (also keine Rundung wie bei CINT).

Verwandte Befehle: CINT,INT,ROUND

FOR

FOR <einfache Variable> = <Start> TO <Ende> [STEP <Schrittweite>]

```
FOR DAY=1 TO 5 STEP 2
```

KOMMANDO: Wiederholt einen vorgegebenen Teil eines Programms so oft, wie durch die <Start>- und <Ende>-Werte angegeben wird. Falls keine <Schrittweite> angegeben ist, wird ein Wert 1 angenommen.

FRE

(<Numerischer Ausdruck>)

```
PRINT FRE(0)
PRINT FRE("")
```

FUNKTION: Gibt die Größe des von BASIC nicht benutzten Speicherplatzes zurück. FRE("") löst ein Sammeln auch kleinster freier Speicherplätze aus, bevor die Größe des freien Speicherplatzes zurückgegeben wird.

GOSUB

GOSUB <Zeilennummer>

```
GOSUB 210
```

KOMMANDO: Ruft ein BASIC-Unterprogramm, indem zur angegebenen <Zeilennummer> gesprungen wird. (Siehe RETURN)

Verwandte Befehle: RETURN

GOTO

GOTO <Zeilennummer>

```
GOTO 90
```

KOMMANDO: Springt zur angegebenen <Zeilennummer>.

HEX$

HEX$ (<Vorzeichenloser Ganzzahliger Ausdruck>)[,<Ganzzahliger Ausdruck>]

```
PRINT HEX$(65534)
FFFE
```

FUNKTION: Gibt den Wert des <vorzeichenlosen ganzzahligen Ausdrucks> in hexadezimaler Form zurück (siehe Anhang II). Der zweite <ganzzahlige Ausdruck> gibt die Zahl der zurückzugebenden Zeichen an und muß zwischen 0 und 16 liegen.

Verwandte Befehle: BIN$,STR$

HIMEM

HIMEM

```
?HIMEM
 42903
```

VARIABLE: Enthält die höchste von BASIC belegte Speicheradresse und kann in numerischen Ausdrücken in üblicher Weise benutzt werden.

Bevor man mit dem MEMORY-Kommando die oberste Speicheradresse neu festlegt, empfiehlt es sich, mit dem Kommando mm=HIMEM den bisherigen Wert sicherzustellen, um ihn später mit MEMORY mm wieder setzen zu können.

Verwandte Befehle: FRE, MEMORY

IF

IF<logischer Ausdruck> THEN <Befehle> [ELSE <Befehle>]

```
IF A>B THEN A=C ELSE A=D
IF A>B GOTO 1000 ELSE 300
```

KOMMANDO: Je nach dem Ergebnis des <logischen Ausdrucks> wird entweder die eine oder die andere Anweisung ausgeführt. Ergibt der <logische Ausdruck> den Wert 'wahr', wird der THEN- oder GOTO-Teil ausgeführt. Im Falschfalle wird der ELSE-Teil ausgeführt oder, falls keiner angegeben ist, mit dem nächsten gefundenen Befehl weitergemacht. Ein IF-Kommando kann nicht länger als eine Zeile sein und darf als <Befehle> beliebige Kommandos beinhalten, außer ein weiteres IF-Kommando.

Verwandte Befehle: THEN,ELSE,GOTO,OR,AND,WHILE

INK

INK <Farbstift>, <Farbe>[,<Farbe>]

```
INK 0,1
```

KOMMANDO: In Abhängigkeit vom augenblicklichen Bildschirmmodus können einer Anzahl von <Farbstiften> <Farben> zugeordnet werden (siehe Kapitel 5). Die <Farbe> oder die <Farben>, die ein >Farbstift> benutzen soll, können durch das INK-Kommando gemäß der Farbtabelle in Anhang IV geändert werden.

Verwandte Befehle: PEN,PAPER

INKEY

INKEY(<Ganzzahliger Ausdruck>)

```
10 CLS:IF INKEY(55)=32 THEN 30 ELSE 20
20 CLS:GOTO 10
30 PRINT "Sie haben [SHIFT] und V gedrueckt"
40 FOR t=1 TO 1000:NEXT:GOTO 10
```

FUNKTION: Gibt zurück, welche Taste der Tastatur gedrückt wurde. Die Tastatur wird in Abständen von 0,02 Sekunden abgefragt. Die Funktion kann benutzt werden, um Ja/Nein-Abfragen auszuwerten, wenn es dabei nicht wichtig ist, zu wissen, ob große oder kleine Buchstaben eingegeben wurden. Das oben angegebene Beispiel prüft, ob [SHIFT] und V gleichzeitig gedrückt wurden. Die zurückgegebenen Werte der Funktion in Abhängigkeit von den gedrückten Tasten sind

Rückgabewert	[SHIFT]	[CTRL]	Taste
–1	?	?	nicht gedrückt
0	nicht gedrückt	nicht gedrückt	gedrückt
32	gedrückt	nicht gedrückt	gedrückt
128	nicht gedrückt	gedrückt	gedrückt
160	gedrückt	gedrückt	gedrückt

Verwandte Befehle: INPUT,INKEY$

INKEY$

INKEY$

```
10 CLS
20 PRINT "Sind Sie klug (j oder n)?"
30 a$=INKEY$: IF a$="" GOTO 30
40 IF a$="N" OR a$="n" THEN
   PRINT "Sie muessen es gewesen sein, als Sie mich
   gekauft haben!":END
50 IF a$="J" OR a$="j" THEN
   PRINT "Sie sind zu bescheiden!!!":End
60 GOTO 20
```

FUNKTION: Gibt den Wert der gedrückten Taste zurück, um Eingaben übernehmen zu können. Ohne daß die [ENTER]-Taste bei jeder Antwort gedrückt wird, wird das entsprechende Zeichen zurückgegeben. Andernfalls wird ein 'Nulltext' zurückgegeben, der für eine Programmschleife verwendet werden kann, um darauf zu warten, bis wirklich eine Eingabe gemacht wird.

Verwandte Befehle: INPUT,INKEY

INP

INP(<Schnittstellenadresse<)

```
PRINT INP(&FF77)
```

FUNKTION: Gibt den Eingabewert der angegebenen Eingabeschnittstelle (I/O port) zurück:

Verwandte Befehle: OUT,WAIT

INPUT

INPUT[#<Ein-/Ausgabenummer>,][;][<Ausgabetext>;]<Liste von: [Variablen]>
INPUT [#<Ein-/Ausgabenummer>,][;][<Ausgabetext>,]<Liste von: [Variblen]>

```
10 CLS
20 INPUT "Geben Sie zwei durch Komma getrennte
   Zahlen ein ";A,B
30 IF A=B THEN PRINT "Die beiden Zahlen sind gleich"
40 IF A>B THEN PRINT A "ist groesser als" B
50 IF A<B THEN PRINT A "ist kleiner als" B
60 CLEAR:GOTO 20
```

KOMMANDO: Liest Daten vom mit <Ein-/Ausgabenummer> angesprochenen Eingabegerät. Ein Strichpunkt nach INPUT unterdrückt den Zeilenvorschub nach der Eingabe. Ein Strichpunkt nach dem <Ausgabetext> gibt als Eingabesignal ein Fragezeichen aus. Ein Komma unterdrückt das Fragezeichen. Wird eine Eingabe gemacht, die nicht zum angegebenen Variablentyp paßt (z. B. der Buchstabe O statt die Zahl 0 für einen <numerischen Ausdruck>), antwortet BASIC mit

```
?REDO from start
```

und der Wiederholung des <Ausgabetextes> und wartet auf eine neue Eingabe.

Alle Eingaben müssen mit [ENTER] abgeschlossen werden. Der Strichpunkt nach <Ein-/Ausgabenummer> unterdrückt den Zeilenvorschub, d. h. der Cursor bleibt hinter dem letzten eingegebenen Zeichen in der Zeile stehen. Handelt es sich um eine Eingabe von Cassette, so wird der <Ausgabetext> unterdrückt. So kann dasselbe INPUT-Kommando durch Umsetzen der Variablen <Ein-/Ausgabenummer> mit unterschiedlichen Auswirkungen verwendet werden.

Je ein Datenausdruck wird für jede Variable in der <Liste von:<Variablen>> aus der Datei gelesen. Die Eingaben müsen mit den Variablentypen im INPUT Kommando übereinstimmen und die einzelnen Eingaben durch ein Komma, [ENTER], Leertaste oder Dateiende getrennt bzw. abgeschlossen werden. Kommas oder [ENTER]-Eingaben nach Leerzeichen werden ignoriert. Texte, die nicht in Anführungszeichen stehen, werden mit Komma oder Leertaste in Einzelwerte aufgeteilt.

Verwandte Befehle: LINE INPUT,READ,INKEY$

INSTR

INSTR[(<Ganzzahliger Ausdruck>,]<Textausdruck>,<Textausdruck>)

```
PRINT INSTR(2,"BANANA","AN")
```

FUNKTION: Gibt die Positionsnummer zurück, an der der zweite <Textausdruck> gefunden wurde. Die Suche beginnt am Anfang des ersten <Textausdrucks> oder bei der durch den <Ganzzahligen Ausdruck> angegebenen Position.

Verwandte Befehle: MID$,LEFT$,RIGHT$

INT

INT(<Numerischer Ausdruck>)

```
PRINT INT(-1.995)
-2
```

FUNKTION: Gibt den zum nächstkleineren gerundeten ganzzahligen Wert zurück. INT arbeitet wie das FIX-Kommando, nur daß für negative Werte um 1 kleinere Werte zurückgegeben werden, falls sie nicht schon ganzzahlig sind.

Verwandte Befehle: CINT,FIX,ROUND

JOY

JOY(<ganzzahliger Ausdruck>)

```
10 IF JOY(0)=8 THEN GOTO 100
```

FUNKTION: Gibt den Wert der Eingabe vom durch den <Ganzzahligen Ausdruck> (entweder 0 oder 1) angegebenen Joystick zurück. Dabei werden die Bewegungen durch die Werte der einzelnen Bits weitergegeben.

Bit	Bedeutung	Dezimalwert
0:	Vorwärts	1
1:	Rückwärts	2
2:	Links	4
3:	Rechts	8
4:	Feuer 2	16
5:	Feuer 1	32

Verwandte Befehle: INKEY

KEY

KEY <ganzzahliger Ausdruck>, <Textausdruck>

```
KEY 140,"RUN"+CHR$(13)
```

KOMMANDO: Belegt eine Funktionstaste mit Werten. Wie in Anhang III beschrieben, gibt es 32 Tastatur-'erweiterungs'-zeichen im Bereich von 128 bis 159. Immer wenn eine dieser Tasten gedrückt wird, so wird der bis zu 32 Zeichen lange zugeordnete Text ausgeführt. Die Summe aller gespeicherten 'Erweiterungs'-texte kann 100 Zeichen nicht übersteigen. Das KEY-Kommando ordnet den angegebenen <Textausdruck> der durch den <ganzzahligen Ausdruck> angesprochenen Taste zu.

Verwandte Befehle: KEY DEF

KEY DEF

KEY DEF <Tastennummer>,<Wiederholung>[,<Normalzeichen>[,<mit Shift>[,<mit CTRL>]]]

```
KEY DEF 46,1,63
```

KOMMANDO: Verändert den einer Taste zugeordneten Wert (siehe Anhang III). Das obige Beispiel führt dazu, daß beim Drücken der N-Taste das ? (Zeichen 63) ausgegeben wird. Um der Taste ihre ursprüngliche Bedeutung wieder zu geben, schreibt man:

```
KEY DEF 46,1,110
```

wobei das Zeichen 110 das n bedeutet.

Verwandte Befehle: KEY

LEFT$

LEFT$(<Textausdruck>,<ganzzahliger Ausdruck>)

```
10 CLS
20 A$ = "AMSTRAD"
30 B$ = LEFT$(A$,3)
40 PRINT B$
RUN
    [Löschen des Bildschirms]
SCH
Ready
```

FUNKTION: Gibt aus dem <Textausdruck> vom Beginn an soviele Zeichen zurück, wie der <ganzzahlige Ausdruck> angibt. Ist der <Textausdruck> kürzer als die gewünschte Länge, so wird der gesamte <Textausdruck> zurückgegeben.

Verwandte Befehle: MID$,RIGHT$

LEN

LEN(<Textausdruck>)

```
A$= "AMSTRAD" :PRINT LEN(A$)
```

FUNKTION: Gibt die Anzahl aller Zeichen einschließlich Leerzeichen zurück, die der <Textausdruck> enthält.

LET

LET<logischer Ausdruck>
LET x=100

KOMMANDO: Veraltete Form aus früheren BASIC-Sprachen, Variablen einen Wert zuzuweisen. Der Befehl ist aus Kompatibilität zugelassen. Das Beispiel muß im Amstrad BASIC nur in der Form.

```
x=100
```

geschrieben werden.

LINE INPUT

LINE INPUT [#<Ein-/Ausgabegerät>,][;][<Ausgabetext>;]<Textvariable>
LINE INPUT [#<Ein-/Ausgabegerät>,][;][<Ausgabetext>,]<Textvariable>

```
LINE INPUT A$
LINE INPUT "NAME";N$
```

KOMMANDO: Liest eine Zeile vom angegebenen <Ein-/Ausgabegerät>. Ein ; nach LINE INPUT unterdrückt den Zeilenvorschub nach der Eingabe. Ist kein <Ein-/Ausgabegerät> angegeben, so wird standardmäßig #0, die Bedienkonsole angenommen.

Verwandte Befehle: READ,INPUT,INKEY$,INPUT$

LIST

LIST [<Zeilenbereich>][,#<Ein-/Ausgabegerät>]

```
LIST 100-1000,#1
```

KOMMANDO: Listet die angegebenen Zeilen auf dem angesprochenen <Ein-/Ausgabegerät> auf. #0ist die Standard-Bildschirmausgabe, #8 der Drucker. Die Auflistung kann durch einmaliges Drücken der [ESC]-Taste unterbrochen und durch die Leertaste wieder fortgesetzt werden. Zweimaliges Drücken von [ESC] bricht die Ausgabe ab. Programme können in vorher definierten Bildschirmteilbereichen (WINDOW) aufgelistet werden, um z. B. Fehler suchen zu können, ohne den anderen Bildschirminhalt zu überschreiben. Die Auflistung vom Beginn des Programms bis zu einer bestimmten Zeile oder von einer bestimmten Zeile bis zum Programmende kann auch unter Weglassen der ersten bzw. letzten Zeilenzahl des <Zeilenbereichs> veranlaßt werden, z. B.

```
LIST-200   oder   LIST 30-
```

LOAD

LOAD <Dateiname>[<Adressenausdruck>]

```
LOAD "Programm 1"
```

KOMMANDO: Liest ein BASIC-Programm von Cassette in den Speicher und löscht dabei ein eventuell schon vorhandenes oder, falls der <Adressenausdruck> angegeben ist, wird eine binäre Datei in den Speicher geladen (siehe Kapitel 2).

LOCATE

LOCATE [#<Ein-/Ausgabegerät>,]<x-Koordinate>,<y-Koordinate>

```
10 MODE 1
20 LOCATE 20,12
30 PRINT CHR$(249)
```

KOMMANDO: Positioniert den Cursor auf die angegebene x/y-Koordinate, die relativ zum Ursprung des Bildschirmteilbereichs (WINDOW) ausgewertet wird. Die linke obere Ecke eines Bildschirmteilbereichs hat die Koordinate 1/1. Ist kein <Ein-/Ausgabegerät> angegeben, so wird #0 angenommen.

Verwandte Befehle: WINDOW

LOG

LOG(<numerischer Ausdruck>)

```
?LOG(9999)
 9.21024037
```

FUNKTION: Gibt den Wert des natürlichen Logarithmus des <numerischen Ausdrucks> als reelle Zahl (mit Dezimalpunkt) zurück.

Verwandte Befehle: EXP,LOG10

LOG10

LOG10(<numerischer Ausdruck>)

```
?LOG10(9999)
 3.99995657
```

FUNKTION: Gibt den Wert des Logarithmus zur Basis 10 des <numerischen Ausdrucks> als reelle Zahl (mit Dezimalpunkt) zurück.

Verwandte Befehle: EXP,LOG

LOWER$

LOWER$(<Textausdruck>)

```
A$= "AMSTRAD" :PRINT LOWER$(A$)
schneider
```

FUNKTION: Gibt den angegebenen <Textausdruck> von Großbuchstaben – oder gemischter Schreibweise umgewandelt in Kleinbuchstabenschreibweise zurück. Ein nützliches Hilfsmittel, um Eingaben auszuwerten, die in unterschiedlicher Schreibweise eingegeben sein können.

Verwandte Befehle: UPPER$

MAX

MAX(<Liste von:<numerischen Ausdrücke>)

```
10 n=66
20 PRINT MAX(1,n,3,6,4,3)
```

FUNKTION: Gibt den größten Wert aus der Liste der <numerischen Ausdrücke> zurück.

Verwandte Befehle: MIN

MEMORY

MEMORY <Adressenausdruck>

```
MEMORY &20AA
```

KOMMANDO: Setzt die Obergrenze des BASIC zur Verfügung stehenden Speichers (siehe Beschreibung des Kommandos HIMEM). Um den Speicherbereich zu überprüfen verwendet man das FRE-Kommando.

Verwandte Befehle: HIMEM,FRE

MERGE

MERGE [<Dateiname>]

```
MERGE "PLAN"
```

KOMMANDO: Mischt ein Programm von Cassette in das Programm im Speicher. Der Inhalt der Datei wird dem Programm im Speicher hinzugefügt. Falls kein <Dateiname> angegeben ist, wird das erste auf der Cassette gefundene Programm genommen. Falls vor dem Dateinamen ein ! steht, so werden die üblichen Meldungen im Zusammenhang mit dem Cassettenlesen unterdrückt. Falls ein Programm in das andere gemischt werden soll, ohne Zeilen des schon im Speicher befindlichen zu überschreiben, so sollte letzteres in einen Zeilenbereich umnummeriert (RENUM) werden, der über dem des einzulesenden liegt.

Alle Variablen, benutzerdefinierte Funktionen und offene Dateien werden gelöscht, ein ON ERROR GOTO wird abgeschaltet, ein RESTORE wird durchgeführt und die DEFINT,DEFREAL- und DEFSTR-Festlegungen verlieren ihre Wirkung. Geschützte Dateien können nicht mit dem MERGE- Kommando gelesen werden.

Verwandte Befehle: LOAD,CHAIN,CHAIN MERGE

MID$

MID$(<Textausdruck>,<ganzzahliger Ausdruck>[,<ganzzahliger Ausdruck>])

```
A$="SCHNEIDER":PRINT MID$(A$,2,4)
CHNE
```

FUNKTION: Gibt ab der mit dem ersten <ganzzahligen Ausdruck> angegebenen Position aus dem <Textausdruck> den Text zurück. Mit dem zweiten <ganzzahligen Ausdruck> wird die Anzahl der zu extrahierenden Zeichen angegeben (Standard: Bis zum Ende des <Textausdrucks>).

Verwandte Befehle: LEFT$,RIGHT$

MIN

MIN(<Liste von <numerischen Ausdrücke>>)

```
PRINT MIN(3,6,2.999,8,9)
 2.999
```

FUNKTION: Gibt den kleinsten Wert aus der Liste der <numerischen Ausdrücke> zurück.

Verwandte Befehle: MAX

MODE

MODE<ganzzahliger Ausdruck>

```
MODE 1
```

KOMMANDO: Ändert den Bildschirmmodus (0,1 oder 2) und löscht den Bildschirm mit INK 0. Alle Bildschirmteilbereiche (WINDOW) werden auf Bildschirmgröße und der Cursor auf den Nullpunkt gesetzt.

Verwandte Befehle: WINDOW,ORIGIN

MOVE

MOVE <x-Koordinate>,<y-Koordinate>

```
MOVE 34,34
```

KOMMANDO: Positioniert den Graphik-Cursor auf die angegebene absolute x/y-Koordinate.

Verwandte Befehle: MOVER,PLOT,PLOTR,DRAW,DRAWR,ORIGIN,TEST,
TESTR,XPOS,YPOS

MOVER

MOVER <x-Versatz>,<y-Versatz>

```
MOVER 34,34
```

KOMMANDO: Positioniert den Graphik-Cursor von seiner augenblicklichen Position auf eine x/y-Koordinate, die sich aus dem relativen x/y-Versatz ergibt.

Verwandte Befehle: MOVE,PLOT,PLOTR,DRAW,DRAWR,TEST,TESTR,
XPOS,YPOS

NEW

NEW

```
NEW
```

KOMMANDO: Löscht das sich im Speicher befindende Programm und seine Variablen. Tastendefinitionen bleiben erhalten, der Bildschrim wird nicht gelöscht. Das Kommando entspricht fast einem ganzen Rücksetzen des Rechners oder dem Ausschalten. Es ist deshalb mit Vorsicht zu benützen.

NEXT

NEXT [<Liste von <Variablen>>]

```
FOR n=1 TO 1000:NEXT
```

KOMMANDO: Legt das Ende einer FOR-Schleife fest. Das NEXT-Kommando kann ohne oder mit dem Variablennamen angegeben werden, der auf das entsprechende FOR-Kommando verweist. In obigem Beispiel hätte das Kommando dann die Form

```
NEXT n
```

Verwandte Befehle: FOR

ON GOSUB
ON GOTO

ON <ganzzahliger Ausdruck> GOSUB <Liste von <Zeilennummern>>
ON <ganzzahliger Ausdruck>GOTO <Liste von <Zeilennummern>>

```
10 ON tag GOSUB 100,200,300,400,500
10 ON rate GOTO 1000,2000,3000,4000
```

KOMMANDO: Springt in Abhängigkeit vom <ganzzahligen Ausdruck> in ein Unterprogramm (GOSUB) oder zu einer Zeile im Programm (GOTO). Falls der Wert des <ganzzahligen Ausdrucks> 1 ist, wird zur ersten Zeilennummer, bei 2 zur zweiten usw. gesprungen. in obigem Beispiel wird bei tag=1 zum Unterprogramm bei Zeile 100, bei tag=2 zum Unterprogramm bei Zeile 200 usw. gesprungen.

Verwandte Befehle: GOTO,GOSUB

ON BREAK GOSUB

ON BREAK GOSUB <Zeilennummer>

```
10 ON BREAK GOSUB 40
20 PRINT "Programm arbeitet"
30 GOTO 20
40 CLS
50 PRINT "Zweimal [ESC] ruft das Unterprogramm"
60 FOR t=1 TO 2000:NEXT
70 RETURN
```

KOMMANDO: Springt in ein Unterprogramm auf <Zeilennummer>, wenn zweimal die [ESC]-Taste gedrückt wird.

Verwandte Befehle: ON BREAK STOP,RETURN

ON BREAK STOP

ON BREAK STOP

```
10 ON BREAK GOSUB 40
20 PRINT "Programm arbeitet"
30 GOTO20
40 CLS
50 PRINT "Zweimal [ESC] ruft das Unterprogramm"
60 FOR t=1 TO 2000:NEXT
65 ON BREAK STOP
70 RETURN
```

KOMMANDO: Stoppt das Programm, wenn zweimal die [ESC]-Taste gedrückt wird. In obigem Beispiel springt das Programm beim ersten Doppel-[ESC] in das Unterprogramm, beim zweiten wird das Programm gestoppt.

Verwandte Befehle: ON BREAK GOSUB

ON ERROR GOTO

ON ERROR GOTO <Zeilennummer>

```
10 ON ERROR GOTO 80
20 CLS
30 PRINT"Falls ich einen Fehler gemacht habe"
40 PRINT"will ich das Programm auflisten, um"
50 PRINT"zu sehen, wo ich den Fehler gemacht habe".
60 FOR t=1 TO 4000:NEXT
70 GOTO 200
80 CLS:PRINT"Fehler in Zeile";ERL:PRINT
90 LIST
```

KOMMANDO: Springt beim Auftreten eines Fehlers zur angegebenen Zeile. In obigem Fall tritt in Zeile 70 ein Fehler auf.

Verwandte Befehle: ERR,ERL,RESUME

ON SQ GOSUB

ON SQ(<Kanal>) GOSUB <Zeilennummer>

ON SQ (2) GOSUB 2000

KOMMANDO: Springt zum Unterprogramm an der angegebenen Zeile, falls bei der Tonausgabe in der Warteschlange des angegebenen <Kanals> Platz frei ist. <Kanal> ist ein <ganzzahliger Ausdruck> mit folgenden Werten und Bedingungen:

1 für Kanal A
2 für Kanal B
4 für Kanal C

Verwandte Befehle: SOUND,SQ

OPENIN

OPENIN <Dateiname>

```
100 OPENIN "!INFORMATION"
```

KOMMANDO: Eröffnet eine Eingabedatei von Cassette. Falls vor dem <Dateinamen> ein ! angegeben wird, werden die üblichen Meldungen der Cassetteneingabe unterdrückt, und das Programm liest den ersten Block mit dem <Dateinamen> von der Cassette.

Verwandte Befehle: CLOSEIN,OPENOUT

OPENOUT

OPENOUT <Dateiname>

```
OPENOUT "!DATEN1"
```

KOMMANDO: Eröffnet eine Ausgabedatei auf Cassette. Falls vor dem <Dateinamen> ein ! angegeben wird, werden die üblichen Meldungen der Cassettenausgabe unterdrückt, und das Programm schreibt den ersten Datenblock mit dem <Dateinamen> auf die Cassette. Jeder Datenblock ist 2K groß. Ein Datenblock wird erst dann wirklich geschrieben, wenn der 2K-Puffer voll ist, oder die Datei mit dem CLOSEOUT-Kommando geschlossen wird.

Verwandte Befehle: CLOSEOUT,OPENIN

ORIGIN

ORIGIN <x>,<y>[,<Links>,<Rechts>,<Oben>,<Unten>]

```
10 CLS:BORDER 13
20 ORIGIN 0,0,50,590,350,50
30 DRAW 540,350
40 GOTO 20
```

KOMMANDO: Bestimmt den Startpunkt des Graphik-Cursors. Der [wahlweise Teil] enthält den Kommandoteil, der einen neuen Graphik-Bildschirmbereich (WINDOW) festlegt, der in allen Bildschirmmodi mit seinen Punkten angesprochen werden kann.

Der Nullpunkt (ORIGIN) ist der Punkt am linken unteren Eck des Bildschirmteilbereichs (WINDOW), die Koordinaten steigen nach rechts und nach oben auf. Liegen Ecken des Bildschirmteilbereichs außerhalb der echten Bildschirmfläche, so werden sie dennoch akzeptiert. Die außerhalb gezeichneten Graphiken sind dann natürlich nicht sichtbar.

Verwandte Befehle: WINDOW

OUT

OUT <Interfaceadresse>,<ganzzahliger Ausdruck>

```
OUT &F8F4,10
```

KOMMANDO: Schickt den Wert des <ganzzahligen Ausdrucks», der zwischen 0 und 255 liegen muß, zur Ausgabe an das mit seiner Adresse angesprochene Interface.

Verwandte Befehle: INP,WAIT

PAPER

PAPER [#<Ein-Ausgabegerät>,]<Farbstift>

```
10 MODE 0
20 FOR p=0 TO 15
30 PAPER p:CLS
40 PEN 15-p
50 LOCATE 6,12:PRINT "PAPER"p
60 FOR t=1 TO 500: NEXT t
70 NEXT p
```

KOMMANDO: Legt fest, welcher Farbstift zum Ausfüllen des beschreibbaren Bildschirmbereichs verwendet werden soll. Wird ein Zeichen auf den Bildschirm geschrieben, so wird der zum Zeichen gehörende Hintergrund mit der dem <Farbstift> (ink) zugeordneten Farbe des PAPER-Kommandos gefüllt, es sei denn, es wurde im 'Transparentmode' ausgegeben.

Die Anzahl der zu verwendenden <Farbstifte> hängt vom ausgewählten Bildschirmmodus ab. Falls der angegebene <Farbstift> nicht zulässig ist, wird ein beliebiger verwendbarer ausgewählt.

Verwandte Befehle: INK,WINDOW,PEN

PEEK

PEEK (<Adressenausdruck>)

```
10 MODE 2
20 INK 1,0: INK 0,12 : BORDER 12
30 INPUT "Startadresse des Speicherbereichs";anfang
40 INPUT "Endadresse des Speicherbereichs";ende
50 FOR n= anfang TO ende
60 VALUE$=HEX$(PEEK(n),2)
70 PRINT VALUE$
80 PRINT" an der Adresse ";HEX$(n,4)
90 NEXT
```

FUNKTION: Gibt den Inhalt der angegebenen Speicheradresse zurück. Mit obigem Beispiel kann man sich den Inhalt des RAMs des CPC464 anschauen. Damit ist es möglich, im RAM-Bereich zu lesen, der vom unteren (von &0000 bis &3FFFF) und vom oberen (von &C000bis &FFFF) Bereich des ROMs überlagert wird, nicht im ROM selbst.

Verwandte Befehle: POKE

PEN

PEN [#<Ein-/Ausgabegerät>]<Farbstift>

PEN 1,2

KOMMANDO: Ordnet dem Schreib- oder Zeichenstift den angegebenen <Farbstift> zu. Ist kein <Ein-/Ausgabegerät> angegeben, so wird standardmäßig #0, der Bildschirm angenommen.

Verwandte Befehle: INK,PAPER

PI

PI

```
PRINT PI
3.14159265
```

```
10 REM Perspektivisches Zeichnen
20 MODE 2
30 RAD
40 INK 1,0
50 INK 0,12
60 BORDER 9
70 FOR N=1 TO 200
80 ORIGIN 420,0
90 DRAW 0,200
100 REM Zeichne Winkel vom Fluchtpunkt
110 DRAW 30*N*SIN(N*PI/4),(SIN(PI/2))*N*SIN(N)
120 NEXT
130 MOVE 0,200
140 DRAWR 0,50
150 DRAWR 40,0
160 WINDOW 1,40,1,10
170 PRINT"Schreiben Sie das Galgen-Programm selbst
    weiter!"
```

FUNKTION: Gibt den Wert von PI zurück, dem Verhältnis zwischen Umfang und Durchmesser des Kreises, nämlich 3.141592653468251. Die Funktion wird in den meisten graphischen Programmen verwendet.

Verwandte Befehle: DEG,RAD

PLOT

PLOT <x-Koordinate>,<y-Koordinate>[,<Farbstift>]

```
10 MODE 2:PRINT "Geben Sie 4 durch Kommas getrennte
   Zahlen ein":PRINT
20 PRINT "Eingabe von x-Mittelpunkt (0-639),
   y-Mittelpunkt (0-399), Radius und Schrittweiten-
   winkel":INPUT x,y,r,s
30 ORIGIN x,y
40 FOR angle = 1 to 360 STEP s
50 XPOINT = r*COS(angle)
60 YPOINT = r*SIN(angle)
70 PLOT XPOINT,YPOINT
74 REM MOVE 0,0
75 REM DRAW XPOINT,YPOINT
80 NEXT
```

KOMMANDO: Plottet jeden Einzelpunkt von der momentanen Cursorposition bis zur angegebenen absoluten x/y-Koordinate. Das **PLOT**-Kommando ist dem **DRAW**-Kommando ähnlich. Versuchen Sie obiges Beispiel mit den Werten 320, 200, 20,1. Entfernen Sie dann das Wort **REM** in der Zeile 75, um den Befehl dieser Zeile zu ermöglichen. In Zeile 70 muß natürlich ein **REM** eingefügt werden. Sie sehen jetzt den Unterschied. (Wenn Sie das **REM** in Zeile 74 entfernen, füllen Sie den Kreis aus.)
Beachten Sie, daß das Programm den Kreis erst in mehreren Umläufen erstellt, da der **RAD**ian-(Bogenmaß-)modus nicht ausgeschaltet wurde, so daß 'angle' bei jedem Schritt mehr als ein Winkelgrad bedeutet. Fügen Sie die Zeile 25 mit einem **DEG**-Kommando ein und lassen Sie das Programm noch einmal laufen.

Verwandte Befehle: **DRAW,DRAWR,PLOT,PLOTR,MOVE,MOVER,ORIGIN, TEST,TESTR,XPOS,YPOS**

PLOTR

PLOTR <x-Versatz>,<y-Versatz>[,<Farbstift>]

```
10 MODE 2:PRINT "Geben Sie 4 durch Kommas getrennte
   Zahlen ein":PRINT
20 PRINT "Eingabe von x -Mittelpunkt(0-639),
   y-Mittelpunkt(0-399), Radius und Schrittweiten-
   winkel":INPUT x,y,r,s
30 ORIGIN x,y
40 FOR angle = 1 to 360 STEP s
50 XPOINT =r*COS(angle)
60 YPOINT = r*SIN(angle)
70 PLOTR XPOINT, YPOINT
80 NEXT:GOTO40
```

KOMMANDO: Plottet jeden Einzelpunkt von der momentanen Cursorposition bis zu der sich aus dem Versatz ergebenden relativen x/y-Koordinate. Das **PLOTR**-Kommando ist dem **DRAWR**-Kommando ähnlich. Versuchen Sie die Werte 320,0,20,1 in obigem Beispiel.

Verwandte Befehle: **DRAW,DRAWR,PLOT,PLOTR,MOVE,MOVER,ORIGIN, TEST,TESTR,XPOS,YPOS**

POKE

POKE <Adressenausdruck>,<ganzzahliger Ausdruck>

```
POKE &00FF,10
```

KOMMANDO: Schreibt den <ganzzahligen Ausdruck>, dessen Wert zwischen 0 und 255 liegen muß, in die angegebene Speicherstelle. Bei der Verwendung des POKE-Kommandos ist Vorsicht geboten, da das System ggf. überschrieben werden kann.

Verwandte Befehle: PEEK

POS

POS(#<*Ein-/Ausgabegerät*>)

```
PRINT POS(#0)
 1
```

FUNKTION: Gibt die augenblickliche Position des angegebenen <Ein-/Ausgabegerätes> zurück. Das <Ein-/Ausgabegerät> muß immer angegeben werden, sonst wird ein Fehler (Syntax error) gemeldet. Bildschirm: Es wird die momentane x-Position des Textcursors relativ zum angegebenen Bildschirmteilbereich (WINDOW) zurückgegeben. Dabei ist die linke obere Ecke des Bildschirmteilbereiches immer 1,1.

Drucker: Es wird die momentane Druckposition des Schreibkopfes zurückgegeben, wobei 1 die erste Druckstelle innerhalb der Zeile bedeutet. Alle Zeichen, deren Wert gemäß ASCII-Tabelle größer als &1F(31) ist, werden gezählt.

Verwandte Befehle: VPOS

PRINT

PRINT [#<Ein-/Ausgabegerät>,][<Ausgabeliste>][<USING<Format>] [Trennzeichen>]

```
PRINT #0,"abc"
```

KOMMANDO: Ausgabe auf das angegebene <Ein-/Ausgabegerät>. Eine ausführliche Beschreibung finden Sie auf Seite 54 dieses Kapitels.

Verwandte Befehle: USING,TAB,SPC

RAD

RAD

RAD

KOMMANDO: Schaltet auf Bogenmaßauswertungsmodus um

Verwandte Befehle: DEG,SIN,COS,TAN,ATN

RANDOMIZE

RANDOMIZE [<numerischer Ausdruck>]

```
10 RANDOMIZE 23
20 PRINT RND(6)
```

KOMMANDO: Setzt den Anfangswert für den Zufallszahlengenerator. In BASIC werden Zufallszahlen in Abhängigkeit von einem gegebenen Startwert erzeugt. Die Zahlenfolge ist immer dieselbe. Mit dem RANDOMIZE-Kommando wird entweder ein neuer Startwert mit dem <numerischen Ausdruck> direkt gesetzt oder die Eingabe des Startwertes erzeugt eine Zufallszahlenfolge, die fast unwiederholbar ist.

Verwandte Befehle: RND

READ

READ <Liste von:<Variablen>>

```
10 FOR X=1 TO 4
20 READ N$
30 PRINT N$
40 DATA Bernhard,Peter,Wulf,Marlene
50 NEXT
```

KOMMANDO: Liest Daten aus DATA-Anweisungen entsprechend der angegebenen <Liste von <Variablen>>. Das RESTORE-Kommando setzt den Zeiger für die zu lesenden Daten wieder auf den ersten DATA-Wert (siehe DATA-Kommando).

Verwandte Befehle: DATA,RESTORE

RELEASE

RELEASE <Kanäle>

```
RELEASE 4
```

KOMMANDO: Hebt den eventuellen Wartezustand der angegebenen Ton-<Kanäle> auf. Der Wert, um die <Kanäle> ansprechen zu können, ist Bitabhängig: Kanal A=Bit 0, Kanal B=Bit 1 und Kanal C=Bit 2. Also steht 4 (binär 0100) für den Kanal C.

Verwandte Befehle: SOUND

REM

REM <Rest der Zeile>

```
10 REM Der hier angegebene Text dient der
20 REM Programmdokumentation
```

KOMMANDO: Fügt Kommentarzeilen in das Programm ein. Jedes nach dem REM-Kommando angegebene Zeichen wird als Kommentar betrachtet (auch der :). Ein einfaches Anführungszeichen (') in einer Zeile hat die Bedeutung: REM, es sei denn es ist in einem <Textausdruck> oder einer DATA-Anweisung angegeben.

REMAIN

REMAIN (<ganzzahliger Ausdruck>)

```
REMAIN (3)
PRINT #6,REMAIN(0);
```

FUNKTION: Gibt die verbleibende Restzeit des durch den <ganzzahligen Ausdruck> (zwischen 0 und 3) angesprochenen Weckers zurück und schaltet ihn zusätzlich ab. Falls der angesprochene Wecker nicht eingeschaltet war, wird 0 zurückgegeben.

Verwandte Befehle: AFTER,EVERY

RENUM

RENUM [<neue Zeilennummer>][,[<alte Zeilennummer>][,<Schrittweite>]]

```
RENUM
RENUM 100,,100
```

KOMMANDO: Numeriert das Programm um. Die <neue Zeilennummer> gibt den Startwert der neuen Numerierung an. Der Standardwert ist 10. Mit der <alten Zeilennummer> kann angegeben werden, ab wo innerhalb des Programms umnumeriert werden soll, um z. B. Platz für neue Zeilen zu schaffen. Die <Schrittweite> bestimmt die Abstände zwischen den Zeilennummern. Der Standardwert ist 10. RENUM aktualisiert alle GOSUB- und GOTO-Kommandos. Falls keine Parameter angegeben werden, wird das Programm wie bei RENUM 10,,10 umnumeriert. Zeilennummern können im Bereich von 1 bis 65535 liegen.

RESTORE

RESTORE [<Zeilennummer>]

```
RESTORE 300
```

```
10 FOR N=1 TO 6
20 READ A$
30 PRINT A$ " ";
40 DATA restored,Daten,kann,man,wieder,lesen
50 NEXT
60 PRINT
70 RESTORE
80 GOTO 10
```

KOMMANDO: Setzt den Zeiger für die zu lesenden DATA-Anweisungen auf die erste im Programm auftretende oder auf die mit der <Zeilennummer> angegebene.

Verwandte Befehle: READ,DATA

RESUME

RESUME[<Zeilennummer>]
oder RESUME NEXT

```
RESUME 300
```

KOMMANDO: Veranlaßt das Programm in der nächsten Zeile oder bei der angegebenen <Zeilennummer> weiterzumachen, falls es auf Grund eines ON ERROR GOTO-Sprungs unterbrochen wurde.

Verwandte Befehle: ON ERROR GOTO

RETURN

RETURN

RETURN

KOMMANDO: Rücksprung aus einem Unterprogramm. BASIC kehrt aus einem Unterprogramm zu dem Befehl zurück, der dem GOSUB-Befehl folgt und macht dort weiter.

Verwandte Befehle: GOSUB,ON x GOSUB,ON SQ GOSUB,AFTER n GOSUB,EVERY n GOSUB,ON BREAK GOSUB

RIGHT$

RIGHT$(<Textausdruck>,<ganzzahliger Ausdruck>)

```
10 CLS
20 A$ = "AMSTRAD"
30 B$ = RIGHT$(A$,3)
40 PRINT b$
RUN
[Bildschirm wird gelöscht]
DER
Ready
```

FUNKTION: Gibt die mit dem <ganzzahligen Ausdruck> angegebene Anzahl von Zeichen vom Ende des <Textausdrucks> her gerechnet, zurück. Falls der <Textausdruck> kürzer ist, als die gewünschte Anzahl, so wird der gesamte <Textausdruck> zurückgegeben.

Verwandte Befehle: MID$,LEFT$

RND

RND[(<numerischer Ausdruck>)]

```
10 RANDOMIZE 23
20 PRINT RND(6)
```

FUNKTION: Gibt die nächste Zufallszahl zurück, welche die nächste in der Folge, eine Wiederholung der letzten oder die erste einer neuen Folge sein kann. Das RANDOMIZE-Kommando in obigem Beispiel sorgt dafür, daß RND(6) immer denselben Wert zurückgibt, nämlich 0.146940658.

RND(0) gibt die gleiche Zufallszahl zurück, die beim letztenmal erzeugt wurde. Ist der <numerische Ausdruck> negativ, so ist die erzeugte Zufallszahlenfolge vorhersehbar.

Verwandte Befehle: RANDOMIZE

ROUND

ROUND(<numerischer Ausdruck>[,<ganzzahliger Ausdruck>])

```
10 x=0.123456789
20 FOR r=9 TO 0 STEP -1:PRINT r,ROUND(x,r):NEXT
25 x=123456789
30 FOR r=0 TO -9 STEP -1
40 PRINT r,ROUND (x,r)
50 NEXT
```

FUNKTION: Gibt den gerundeten Wert des <numerischen Ausdrucks> zurück. Dabei wird entweder nach Dezimalstellen oder nach Potenzen von 10 gerundet, je nach dem <ganzzahligen Ausdruck>. Falls der <ganzzahlige Ausdruck> negativ ist, wird der <numerische Ausdruck> so zu einem ganzzahligen Wert gerundet, daß so viele Stellen vor dem Dezimalpunkt durch eine Null ersetzt werden, wie der Absolutwert des <ganzzahligen Ausdrucks> angibt. Die Stellen hinter dem Dezimalpunkt entfallen.

Verwandte Befehle: INT,FIX,CINT,ABS

RUN

RUN <Textausdruck>

```
RUN "PROGRAMM 1"
```

KOMMANDO: Lädt ein Programm von Cassette und startet es. Ist der <Textausdruck> leer ("""), so lädt BASIC das erste gefundene Programm von der Cassette. Wird vor dem Programmnamen ein ! angegeben, so werden die üblichen Meldungen der Casseteneingabe unterdrückt.
Anmerkung: BASIC löscht ein im Speicher vorhandenes Programm beim Laden von Cassette.

Verwandte Befehle: LOAD

RUN

RUN [<Zeilennummer>]

```
RUN 100
```

KOMMANDO: Startet ein im Speicher befindliches Programm von Beginn oder der angegebenen <Zeilennummer>. Ein unterbrochenes Programm wird endgültig abgebrochen, benutzereigene Funktionen und Variablen werden gelöscht, DEFINT, DEFREAL- und DEFSTR-Vereinbarungen verlieren ihre Gültigkeit und offene Dateien werden abgebrochen, wenn das RUN-Kommando eingegeben wird.

Verwandte Befehle: LOAD

SAVE

SAVE <Dateiname>[,<Dateityp>][,<Binärdateiparameter>]

```
SAVE "PROGRAMM",P
```

KOMMANDO: Schreibt das im Speicher befindliche Programm auf Cassette in eine Datei mit dem angegebenen <Dateinamen>. Als <Dateityp> kann dabei angegeben werden:
,A für Sicherung als ASCII-Datei
,P für Sicherung als geschütztes Programm
,B für Speicherung als Binärdatei (z. B. den Bildschirmbereich).
Dabei beschreiben die <Binärdateiparameter> die Startspeicheradresse und die Länge.

Verwandte Befehle: LOAD,RUN "<Dateiname>",MERGE,CHAIN,
CHAIN MERGE

SGN

SGN(<numerischer Ausdruck>)

```
10 INPUT "Wie ist Ihr augenblicklicher Kontostand";
   CASH
20 IF SGN(CASH) <1 GOTO 30 ELSE 40
30 PRINT "Oh je, oh je":END
40 PRINT "Sie haben mehr Geld als ich"
```

FUNKTION: Gibt −1 zurück, falls der <numerische Ausdruck> kleiner 0 ist, 0, falls der <numerische Ausdruck> gleich 0 ist und 1, falls der <numerische Ausdruck> größer ist.

Verwandte Befehle: ABS

SIN

SIN (<numerischer Ausdruck>)

```
PRINT SIN(PI/2)
 1
```

FUNKTION: Gibt den Wert des Sinus des <numerischen Ausdrucks> zurück, der als im Bogenmaß angegeben verstanden wird, solange nicht durch das DEG-Kommando auf Gradmaßinterpretation umgeschaltet wird.

Verwandte Befehle: COS,TAN,ATN,DEG,RAD

SOUND

SOUND <Kanal-Status>,<Tonperiode>[,<Dauer>[,<Lautstärke> [,
<Lautstärkenvariation>[,<Tonvariation>[,<Geräuschfolge>]]]]]

```
SOUND 1,200,1000,7,0,0,1
```

KOMMANDO: Erzeugt Ton. Die Musikmöglichkeiten des CPC464 sind eine der umfassendsten Erweiterungen des BASIC und in Kapitel 6 näher beschrieben.

Verwandte Befehle: ENV,ENT

SPACE$

SPACE$(<ganzzahliger Ausdruck>)

```
SPACE$(190)
```

FUNKTION: Gibt einen <Textausdruck> zurück, gefüllt mit der durch den <ganzzahligen Ausdruck> angegebenen Anzahl von Leerzeichen.

Verwandte Befehle: PRINT,SPC,TAB

SPEED INK

SPEED INK<ganzzahliger Ausdruck>,<ganzzahliger Ausdruck>

```
5 INK 0,9,12:INK 1,0,26
10 BORDER 12,9
20 SPEED INK 50,20
```

KOMMANDO: Setzt die Länge der Farbdauer bei Farbwechseln für INK und BORDER. Das INK- und das BORDER-Kommando erlauben es, je zwei sich abwechselnde Farben anzugeben. Der erste <ganzzahlige Ausdruck> gibt die Dauer der ersten und der zweite <ganzzahlige Ausdruck> der zweiten Farbe an. die Zeiten zwischen den Farbwechseln werden in Einheiten zu 0.02 Sekunden gemessen. Gehen Sie vorsichtig mit dem Auswählen von Farben und Farbwechseln um, da Sie hypnotische Effekte erzielen können.

Verwandte Befehle: INK,BORDER

SPEED KEY

SPEED KEY <Startverzögerung>,<Wiederholungsperiode>

SPEED KEY 20,3

KOMMANDO: Setzt die Werte, entsprechend denen beim dauernden Drücken einer Taste des CPC464 diese wiederholt wird. Nach der angegebenen <Startverzögerung> wird die Taste wie durch die <Wiederholungsperiode> angegeben in gewissen Zeitabständen automatisch wiederholt. Die Zeiten werden dabei in Einheiten von 0,02 Sekunden gemessen. Die angegebenen Werte müssen dabei zwischen 1 und 255 liegen. Die Standardwerte sind jeweils 10.

Eine sehr kleine <Startverzögerung> kann mit den Tastaturmöglichkeiten kollidieren, da die Software nicht die Hardwarefähigkeit beeinflußen kann, mit einer bestimmten 'Geschwindigkeit' eine Taste zu 'lesen'.
Nicht alle Tasten sind als wiederholbar gesetzt. Das KEY DEF-Kommando ermöglicht es dem Benutzer, die Wiederholbarkeit für eine Taste ein- und auszuschalten.

Verwandte Befehle: KEY DEF

SPEED WRITE

SPEED WRITE <ganzzahliger Ausdruck>

SPEED WRITE 1

KOMMANDO: Setzt die Schreibgeschwindigkeit für die Ausgabe auf Cassette. Es kann entweder mit 2000 Baud (<Ganzzahliger Ausdruck> = 1) oder 1000 Baud (<ganzzahliger Ausdruck> = 0) geschrieben werden. Wird eine Datei von Cassette gelesen, schaltet der CPC464 automatisch auf die für die Datei richtige Lesegeschwindigkeit um.

Werden Cassetten minderer Qualität verwendet, sollte zur Erhöhung der Datensicherheit mit 1000 Baud geschrieben werden. Weitere Informationen finden Sie in Kapitel 2.

Verwandte Befehle: SAVE

SQ

SQ (<Kanal>)

```
10 MODE 1
20 FOR n=20 TO 0 STEP-1
30 PRINT n;
40 SOUND 1,10+n,100,7
50 WHILE SQ(1)>127:WEND
60 NEXT
```

FUNKTION: Gibt die Anzahl der freien Plätze in der Tonwarteschlange und den Status der Warteschlange für den angegebenen <Kanal> zurück. Dabei bedeutet 1 den Kanal A, 2 den Kanal B und 3 den Kanal C. Die Funktion gibt zurück, ob der angegebene <Kanal> aktiv ist, und falls nicht, warum der eventuelle erste Eintrag der Warteschlange in einem Wartezustand ist. Die Rückgabe erfolgt in den einzelnen Bits:

Bits 0,1,2: Anzahl der freien Einträge in der Warteschlange
Bits 3,4,5: ggf. den Rendezvousstatus des ersten Eintrags der Warteschlange (Warten auf anderen Kanal)
Bit 6: wird gesetzt, wenn die Warteschlange sich in einem Haltezustand befindet.
Bit 7: wird gesetzt, wenn der Kanal aktiv ist.

Verwandte Befehle: SOUND',ON SQ GOSUB

SQR

SQR(<numerischer Ausdruck>)

```
PRINT SQR(9)
 3
```

FUNKTION: Gibt den Wert der Quadratwurzel aus dem <Numerischen Ausdruck> zurück.

STOP

STOP

```
300 IF n<0 THEN STOP
```

KOMMANDO: Stoppt ein Programm und läßt BASIC in einem Unterbrechungszustand, so daß das Programm wieder fortgesetzt werden kann. Das Kommando kann dazu benutzt werden, um ein Programm zu Testzwecken an einer bestimmten Stelle zu unterbrechen.

Verwandte Befehle: CONT,END

STR$

STR$(<Numerischer Ausdruck>)

```
PRINT STR$(&766)
 1894

PRINT STR$(&X1010100)
 84
```

FUNKTION: Gibt den Dezimalwert des angegebenen <Numerischen Ausdrucks> als Textausdruck zurück in derselben Form, wie er im PRINT-Kommando ausgegeben wird.

Verwandte Befehle: VAL,PRINT,HEX$,BIN$

STRING$

STRING$(<Ganzzahliger Ausdruck>,<Textzeichen>)

```
PRINT STRING$(&16,"*")
 ****************
```

FUNKTION: Gibt einen Textausdruck zurück, der das angegebene <Textzeichen> so oft enthält, wie der angegebene <Ganzzahlige Ausdruck> angibt.

Verwandte Befehle: SPACE$

SYMBOL

SYMBOL <Zeichennummer>,<Liste von: <Beschreibungswerten>>

```
5 MODE 2
10 SYMBOL AFTER 90
20 SYMBOL 93,&80,&40,&20,&10,&8,&4,&2,&1
30 FOR n=1 TO 2000
40 PRINT CHR$(93);
50 NEXT
60 GOTO 60
```

KOMMANDO: Belegt das durch <Zeilennummer> angesprochene Zeichen mit einem neuen Darstellungswert. Die Neubelegung muß vorher mit dem SYMBOL AFTER-Kommando ermöglicht werden. Die <Zeichennummer> muß zwischen 0 und 255 liegen und kann ein ASCII-Zeichen oder eines aus dem CPC464-Zeichensatz ansprechen. Die <Beschreibungswerte> beschreiben bitabhängig von oben nach unten die 8x8 Punktmatrixbelegung des neuen Zeichens. Eine 0 in einem Bit läßt die Schreibflächenhintergrundfarbe, eine 1 die aktuelle Schreibstiftfarbe erscheinen (Siehe auch Anhang II und III). Das obige Beispiel erzeugt einen Schrägstrich von links oben nach rechts unten für die]-Taste.

Verwandte Befehle: SYMBOL AFTER

SYMBOL AFTER

SYMBOL AFTER <ganzzahliger Ausdruck>

```
SYMBOL AFTER 90
```

KOMMANDO: Setzt den Bereich, in dem benutzereigene Zeichen beschrieben werden können. Der Standardwert ist 240, so daß 16 Zeichen beschrieben werden können. Falls der <ganzzahlige Ausdruck> den Wert 32 hat, können alle Zeichen mit den Nummern 32 bis 255 umgesetzt werden.

Wird das SYMBOL AFTER-Kommando gegeben, so werden alle benutzereigenen Zeichenbeschreibungen auf die Standardbelegung zurückgesetzt.

Verwandte Befehle: Symbol

TAG

TAG [#<Ein-/Ausgabegerät>]

```
10 MODE 2
11 BORDER 9
14 INK 0,12
15 INK 1,0
20 FOR n=1 TO 100
30 MOVE 200+n,320+n
40 TAG
50 IF n<70 GOTO 60 ELSE 70
60 PRINT"Auf";:GOTO 80
70 PRINT"Wiedersehen";
80 NEXT
90 GOTO 20
```

KOMMANDO: Ermöglicht mit dem PRINT-Kommando auf Graphikcursorpositionen auszugeben. Damit ist es möglich, Texte und Symbole mit Graphik beliebig zu mischen. Das <Ein-/Ausgabegerät> ist standardmäßig #0, also der Bildschirm. Die linke obere Ecke des Textzeichens wird auf die Graphikcursorposition ausgegeben. Nicht abdruckbare Kontrollzeichen, wie z. B. neue Zeile (LINE FEED) werden als Sonderzeichen ausgegeben, falls hinter dem PRINT-Kommando kein ; steht.

Verwandte Befehle: TAGOFF

TAGOFF

TAGOFF[#<Ein-/Ausgabegerät>]

```
TAGOFF #0
```

KOMMANDO: Schaltet die PRINT-Kommando-Ausgabe auf Graphikcursorpositionen ab. Danach gibt das PRINT-Kommando ab der Textcursorposition weiter aus, die vor dem TAG-Kommando erreicht war.

Verwandte Befehle: TAG

TAN

TAN(<Numerischer Ausdruck>)

```
PRINT TAN(45)
```

FUNKTION: Gibt den Wert des Tangens des <Numerischen Ausdrucks> zurück. Der <Numerische Ausdruck> darf zwischen −200,000 und +200,000 liegen und wird als im Bogenmaß angegeben verstanden, falls nicht auf Gradmaßauswertung umgeschaltet wurde.

Verwandte Befehle: COS,SIN,ATN,DEG,RAD

TEST

TEST(<x-Koordinate>,<y-Koordinate>)

```
PRINT TEST(300,300)
```

FUNKTION: Gibt die Farbstiftnummer zurück, die an der angegebenen absoluten x/y-Koordinate verwendet wurde.

Verwandte Befehle: TESTR,MOVE,MOVER,PLOT,PLOTR,DRAW,DRAWR

TESTR

TESTR(<x-Versatz>,<y-Versatz>)

```
?TESTR(5,5)
```

FUNKTION: Gibt die Farbstiftnummer zurück, die an der durch den x/y-Versatz zur augenblicklichen Graphikcursorposition angegebenen relativen x/y-Koordinate verwendet wurde.

Verwandte Befehle: TEST,MOVE,MOVER,PLOT,PLOTR,DRAW,DRAWR

TIME

TIME

```
10 DATUM = INT(TIME/300)
20 ZEIT = ((TIME/300)-DATUM)
30 PRINT ZEIT
40 GOTO 20
```

FUNKTION: Gibt die Zeit zurück, die seit dem Einschalten vergangen ist, abzüglich der für das Lesen von und Schreiben auf Cassette verbrauchten Zeit. Die Zeit wird in Einheiten zu 1/300 einer Sekunde gemessen.

TRON

TROFF

```
TRON
TROFF
```

```
TRON
```

KOMMANDO: `TRON` ermöglicht es, die Ausführung eines Programms zurückzuverfolgen, da nach dem `TRON`-Kommando vor der Ausführung die jeweilige Zeile in eckigen Klammern `[]` ausgegeben wird. Das `TROFF`-Kommando schaltet die Ausgabe wieder ab.

Verwandte Befehle: `RUN`

UNT

`UNT(`<Adressenausdruck>`)`

```
PRINT UNT(65535)
```

FUNKTION: Gibt den ganzzahligen Wert von −32768 bis +32767 des angegebenen 16-Bit-<Adressenausdrucks> zurück.

Verwandte Befehle: `INT,FIX,CINT,ROUND`

UPPER$

`UPPER$(`<Textausdruck>`)`

```
PRINT UPPER$("schneider")
 SCHNEIDER
```

FUNKTION: Gibt den <Textausdruck> umgewandelt in lauter Großbuchstaben zurück.

Verwandte Befehle: `LOWER$`

VAL

`VAL(`<Textausdruck>`)`

```
10 A$="13 ist meine Schicksalszahl"
20 PRINT VAL (A$)
```

FUNKTION: Gibt einen <Numerischen Ausdruck> zurück, der den Wert der Zahl enthält, die zu Beginn(!) des <Textausdrucks> steht.

Verwandte Befehle: `STR$`

VPOS

VPOS(#<Ein-/Ausgabegerät>)

PRINT VPOS(#0)

FUNKTION: Gibt die vertikale Position (=Zeilennummer) des Textcursors des angegebenen <Ein-/Ausgabegeräts> zurück. Das <Ein-/Ausgabegerät> muß in jedem Fall angegeben werden.

Verwandte Befehle: POS

WAIT

WAIT

WAIT <Interfaceadresse>,<UND-Maske>[,<Exclusive-ODER-Maske>]

WAIT &FF34,20,25

KOMMANDO: Wartet, bis über das angesprochene Interface (Ein-/Ausgabeschnittstelle) ein Wert zwischen 0 und 255 eingelesen ist und weitergegeben wird. Der eingelesene Wert wird ggf. mit der Exklusive-ODER-Maske entsprechend des Exklusiven Oders verknüpft und dann, falls das Ergebnis nicht 0 ist, noch mit der UND-Maske entsprechend der UND-Verknüpfung. BASIC verbleibt solange in einem Wartezustand, bis die Eingabe erfolgt ist. Falls Sie das obige Beispiel eingeben, müssen Sie den Rechner komplett zurücksetzen, um weiterarbeiten zu können.

WEND

WEND

```
10 MODE 1:REM BASIC UHRZEITPROGRAMM
20 INPUT "Geben Sie die aktuelle Uhrzeit ein
   (Stunden,Minuten,Sekunden)";stdn,min,sek
30 CLS:datum = INT(TIME/300)
40 WHILE stdn<24
50 WHILE min<60
60 WHILE ticken<60
70 ticken = (INT(TIME/300)-datum)+sec
80 LOCATE 70,4
90 PRINT #0,USING "##";stdn,min,ticken
100 WEND
110 ticken=0
115 sec=0
120 min=min+1
130 GOTO 30
140 WEND
150 min=0
160 stdn=stdn+1
170 WEND
180 stdn=1
190 GOTO 40
```

KOMMANDO: Schließt eine **WHILE**-Schleife ab. Die **WHILE/END**-Schleife führt Teile des Programms solange aus, bis die entsprechende Bedingung eintritt. Das Beispiel hier verwendet die **WHILE/WEND**-Schleife, um zu zeigen, wie elegant ein Programm durch dieses Leistungsmerkmal gestaltet werden kann.

WHILE

WHILE<Logischer Ausdruck>

```
WHILE TAG <0
```

Siehe **WEND**-Beispiel

KOMMANDO: Setzt den Anfang und startet den Ablauf einer Programmschleife, die solange wiederholt wird, wie der <Logische Ausdruck> zutrifft. Das **WHILE**-Kommando bildet den Anfang der Schleife und definiert die Bedingung. Das **WEND**-Kommando schließt den Schleifenbereich ab.

Verwandte Befehle: **WEND**

WIDTH

WIDTH<ganzzahliger Ausdruck>

```
WIDTH 86
```

KOMMANDO: Begrenzt die Anzahl der Zeichen pro Zeile für die Druckerausgabe. BASIC fügt bei längeren Zeilen automatisch einen Wagenrücklauf und Vorschub auf eine neue Zeile ein.

Verwandte Befehle: PRINT,POS

WINDOW

WINDOW [#<Ein-/Ausgabegerät>,]<Links>,<Rechts>,<Oben>,<Unten>

```
10 MODE 1
20 BORDER 6
30 WINDOW 10,30,7,18
40 PAPER 2:PEN 3
50 CLS
60 PRINT CHR$(143);CHR$(242);"Dies ist die Position"
70 PRINT "1,1 IM TEXTFENSTER"
80 GOTO 80
```

KOMMANDO: Legt einen Bildschirmteilbereich als eigen ansprechbares <Ein-/Ausgabegerät> (stream) fest.

Verwandte Befehle: ORIGIN

WINDOW SWAP

WINDOW SWAP <Ein-/Ausgabegerät>,<Ein-/Ausgabegerät>

```
WINDOW SWAP 0,2
```

KOMMANDO: Tauscht Bildschirmteilbereiche (WINDOW). So können z. B. die auf das <Ein-/Ausgabegerät> #0 üblichen BASIC-Meldungen in einen anderen Bildschirmteilbereich (WINDOW) gelegt werden, um sie von normalen Programmausgaben deutlich unterscheiden zu können.

Verwandte Befehle: WINDOW,PEN,PAPER,TAG

WRITE

WRITE [#<Ein-/Ausgabegerät>,][<Ausgabeliste>]

```
WRITE #2,"HALLO",4,5
"HALLO",4,5
```

KOMMANDO: Gibt die in der <Ausgabeliste> angegebenen Werte durch Kommas getrennt aus, wobei Texte in Doppelanführungszeichen eingeschlossen werden.

Verwandte Befehle: PRINT

XPOS

XPOS

PRINT XPOS

FUNKTION: Gibt die augenblickliche x-Koordinate des Graphikcursors zurück.

Verwandte Befehle: YPOS,MOVE,MOVER,ORIGIN

YPOS

YPOS

PRINT YPOS

FUNKTION: Gibt die augenblickliche y-Koordinate des Graphikcursors zurück.

Verwandte Befehle: XPOS,MOVE,MOVER,ORIGIN

ZONE

ZONE<Ganzzahliger Ausdruck>

```
10 PRINT1,2,3
20 ZONE19
30 PRINT4,5,6
```

KOMMANDO: Ändert die Breite der Druckzonen, die das PRINT-Kommando verwendet. Der Standardwert ist 13, der neue Wert kann zwischen 1 und 255 liegen. Der Zonenwert wird von NEW-, LOAD-, CHAIN- und RUN "<Dateiname>"-Kommandos zurückgesetzt.

Verwandte Befehle: PRINT,WIDTH

PRINT

PRINT[#<Ein-/Ausgabegerät>,][<Ausgabeliste>][USING<Format>]
[<Trennzeichen>]

<Ausgabeliste> steht für <Druckausgabe>[<Trennzeichen>,<Druckausgabe>]*
<Druckausgabe> steht für <Ausdruck>
oder
SPC(<Ganzzahliger Ausdruck>)
oder
TAB(<Ganzzahliger Ausdruck>)

Ausgabe auf Cassette:

```
10 OPENOUT "DATEN"
20 PRINT #9, "Hallo"
30 CLOSEOUT
```

Ausgabe auf Drucker:

```
10 ZAHL=23000*PI
20 PRINT #8, USING "#####.##";ZAHL
30 PRINT #0,ZAHL
RUN
 72256.631
READY
```

[Auf dem Drucker steht]
72256.63

KOMMANDO: Gibt auf das angegebene <Ein-/Ausgabegerät> aus, wobei das angegebene Format verwendet wird (siehe Tabelle der Formatbeschreibungen). Amstrad BASIC erfüllt den gesamten Standard der formatierten PRINT-Ausgabe.

Wird kein Format angegeben, gibt BASIC im 'freien Format' aus. Dabei bedeutet ein , als <Trennzeichen>, daß der nächste auszugebende <Ausdruck> in der nächsten Druckzone beginnt (Standardwert 13 Zeichen) und ein ; als Trennzeichen, daß zwischen den anzugebenden <Ausdrücken> ein Leerzeichen erzeugt wird.

SPC(<Ganzzahliger Ausdruck>) führt zur Ausgabe von Leerstellen entsprechend der durch den <Ganzzahligen Ausdruck> angegebenen Anzahl. Ist die Anzahl kleiner Null, so erfolgt keine Ausgabe, ist sie größer als die zulässige Breite des Ausgabegerätes, so wird sie auf die maximal mögliche Anzahl gekürzt. SPC muß nicht durch ein <Trennzeichen> abgeschlossen werden, da immer ein ; angenommen wird.

TAB(<Ganzzahliger Ausdruck>) gibt Leerzeichen bis zu der durch den <Ganzzahligen Ausdruck> angegebenen Ausgabeposition (Spalte innerhalb einer Zeile). Ist der <Ganzzahlige Ausdruck> negativ, wird 1 angenommen, ist er größer als die maximal mögliche Breite des Ausgabegeräts, so wird er wie bei SPC verkleinert. Ist die gewünschte Ausgabeposition größer als die augenblickliche, so werden bis zur neuen Leerzeichen ausgegeben, ist sie kleiner, so wird bis zum Zeilenanfang zurückgegangen und dann werden von dort aus Leerzeichen ausgegeben, bis die gewünschte Position erreicht ist. Als <Trennzeichen> wird immer ein ; angenommen.

PRINT (Fortsetzung)

```
PRINT USING "<FORMAT>"
```

Für numerische Ausgaben:

Formatzeichen	Beschreibung	Beispiel
#	Jedes # ergibt eine Ausgabestelle	####
.	Position des Dezimalpunkts	#.#
+	Gibt für positive Zahlen das Vorzeichen mit aus	
	entweder zu Beginn	+##
	oder am Ende	##+
-	Gibt für negative Zahlen das Vorzeichen	
	am Ende aus	##-
**	Maximal 2 führende Druckstellen werden mit *	
	statt mit Leerzeichen ausgegeben	**###.##
$$	Vor dem ersten Zeichen wird ein $ ausgegeben	$$##.##
**$	Kombination aus ** und $$	**$#.##
,	Alle drei Stellen vor dem Dezimalpunkt wird eine	
	Zahlengruppe durch ein Komma getrennt	#,###.#
↑ ↑ ↑ ↑	Ausgabe in Exponentialformat. Dabei wird die Zahl	
	so ausgerichtet, daß die erste Stelle nie Null ist.	#.##↑↑↑↑

Für Textausgaben:

!	Nur das erste Zeichen des Textes wird	
	ausgegeben	!
\<Leer-	Soviele Zeichen des Textes wie Leer-	
zeichen>\	zeichen angegeben sind, werden ausgegeben	\ \
&	Der gesamte Textausdruck wird ausgegeben	&

9 Weitergehende Programmierinformationen

Themen dieses Kapitels

* zulässige Textcursorpositionen
* Kontrollzeichen
* Betriebssystem
* Unterbrechungen

9.1 Cursorpositionen und Kontrollzeichen

In Anwenderprogrammen kann es vorkommen, daß der Textcursor außerhalb des augenblicklichen Ausgabebereiches (`WINDOW`) positioniert wird. Verschiedene Ausgabeoperationen führen dazu, daß der Cursor vor ihrer Ausführung auf eine zulässige Position zurückgebracht wird. Dies sind

- Schreiben eines Zeichens
- Ausgeben des Cursors
- Ausführung der Kontrollzeichen, die in der auf den nachfolgenden Seiten angegebenen Liste mit einem Stern gekennzeichnet sind.

Die Rückführung des Cursors auf eine zulässige Position geschieht folgendermaßen:

a) Steht der Cursor außerhalb des rechten Rands, so wird er auf das erste Zeichen der nächsten Zeile gesetzt.

b) Steht der Cursor außerhalb des linken Rands, so wird er auf das letzte Zeichen der vorherigen Zeile gesetzt.

c) Steht der Cursor oberhalb des Ausgabebereichs, so wird der ganze Bereich um eine Zeile nach unten gerollt und der Cursor in die erste Zeile gesetzt.

d) Steht der Cursor unterhalb des Ausgabebereiches, so wird der ganze Bereich um eine Zeile nach oben gerollt und der Cursor in die letzte Zeile gesetzt.

Die Prüfungen und Korrekturen finden in der oben angegebenen Reihenfolge statt. Die zulässigen Cursorpositionen können den Wert 0 haben oder negativ sein, was bedeutet, daß sie links oder oberhalb des Ausgabebereichs liegen.

Zeichenwerte im Bereich von 0 bis 31 (siehe Anhang III) erzeugen kein abdruckbares Zeichen, wenn sie ausgegeben werden, werden aber als Kontrollzeichen interpretiert (sie können bei unvorsichtiger Anwendung den Rechner zum Stillstand bringen). Einige Zeichen ändern die Bedeutung des oder der nachfolgenden Zeichen, da diese dann als Parameter verstanden werden.

Kontrollzeichen, die auf dem graphischen Bildschirm ausgegeben werden, erscheinen als Sonderzeichen (Siehe TAG-Kommando in Kapitel 8). Andere Steuerzeichen, wie z. B. (&Ø7'Klingel'), werden bei Eingabe über Tastatur ([CTRL]G) als Sonderzeichen ausgegeben, führen dagegen ihre zugeordnete Funktion bei der Ausgabe über ein PRINT-Kommando aus (PRINT CHR$(&Ø7).

Die mit * gekennzeichneten Kontrollzeichen führen den Cursor vor ihrer Ausführung auf eine zulässige Position zurück, – können ihn aber danach in einer unzulässigen zurücklassen. Die Kontrollcodes und Ihre Bedeutung sind in der nachfolgenden Liste zuerst mit ihrem hexadezimalen (&XX) und dann mit ihrem Dezimalwert beschrieben.

Zusätzliche Kontrollzeichen-Kommandos: Nicht immer über die Tastatur mit der [CTRL]-Taste eingebbar

Wert	Name	Parameter	Bedeutung
&ØØ Ø	NUL		Kein Einfluß. Ignoriert.
&Ø1 1	SOH	Ø...255	Drucken des als Parameterwert angegebenen Symbols. So können Symbole mit den Werten 0 bis 31 ausgedruckt werden.
&Ø2 2	STX		Textcursor ausschalten.
&Ø3 3	ETX		Textcursor einschalten. Anmerkung: BASIC verhindert das Überschreiben des Cursors; wird nur beim Warten auf die Eingabe über Tastatur aufgehoben.
&Ø4 4	EOT	Ø...2	Bildschirmmode setzen. Parameterauswertung: MOD 4. Gleichwertig mit MODE Kommando.
&Ø5 5	ENQ	Ø...255	Ausgeben des als Parameter angegebenen Symbols auf die Graphikcursorposition.
&Ø6 6	ACK		Textbildschirm aktivieren (siehe &15,NAK, nächste Seite).
&Ø7 7	BEL		Klingel ertönen lassen. Die Tonausgabewarteschlangen werden geleert.
&Ø8 8	* BS		Cursor ein Zeichen zurücksetzen.
&Ø9 9	* TAB		Cursor ein Zeichen vorsetzen.
&ØA 1Ø	* LF		Cursor eine Zeile nach unten setzen.
&ØB 11	* VT		Cursor eine Zeile nach oben setzen.
&ØC 12	FF		Bildschirm löschen und Cursor in die linke obere Ecke setzen. Gleichwertig mit CLS-Kommando.
&ØD 13	* CR		Cursor auf die erste Schreibstelle der aktuellen Schreibzeile setzen.

&0E	14		SO	0...15	Farbstift für den beschreibbaren Bildschirmbereich setzen. Parameterauswertung: MOD 16. Gleichwertig mit PAPER-Kommando.
&0F	15		SI	0...15	Farbstift für die Schreib- und Zeichenausgabe setzen. Parameterauswertung MOD 16. Gleichwertig mit PEN-Kommando.
&10	16	*	DLE		Zeichen auf Cursorposition löschen. Schreibstelle mit Schreibflächenfarbe füllen.
&11	17	*	DC1		Zeile von Beginn bis einschließlich der Cursorposition löschen. Schreibstellen mit aktueller Schreibflächenfarbe füllen.
&12	18	*	DC2		Zeile einschließlich der Cursorposition bis zum Ende löschen. Schreibstellen mit aktueller Schreibflächenfarbe auffüllen.
&13	19	*	DC3		Beginn der Schreibfläche bis einschließlich der Cursorposition löschen. Schreibstellen mit aktueller Schreibflächenfarbe füllen.
&14	20	*	DC4		Einschließlich der Cursorposition bis zum Ende der Schreibfläche löschen. Schreibstellen mit aktueller Schreibflächenfarbe füllen.
&15	21		NAK		Textbildschirm abschalten. Bildschirm reagiert solange nicht mehr, bis ein ACK(06) ausgegeben wird.
&16	22		SYN	0...1	Parameterauswertung: MOD 2; Transparentmodus mit 0 ausschalten, 1 = einschalten.
&17	23		ETB	0...3	Parameterauswertung: MOD 4; Graphikfarbstiftmodus setzen: 0 = Normal 1 = 'XOR''' 2 = 'AND''' 3 = 'XOR'''
&18	24		CAN		Papier- und Schreibstiftfarben austauschen.
&19	25		EM	0...255 0...255 0...255 0...255 0...255 0...255 0...255 0...255 0...255	Matrix für benutzereigene Zeichen setzen. Gleichwertig mit Symbol-Kommando. Benötigt 9 Parameter. Der erste gibt an, welches Zeichen nun zu setzen ist. Die restlichen acht beschreiben die Matrix; das erste (linke) Bit im ersten Byte beschreibt den obersten linken Punkt in der Zeichendarstellung; das letzte (rechte) Bit im letzten Byte stellt den untersten rechten Punkt in der Zeichendarstellung dar.

&1A	26	SUB	1...80 1...80 1...80 1...80	Bildschirmteilbereich (WINDOW) setzen. Gleichwertig mit WINDOW-Kommando. Benötigt 4 Parameter. Die ersten zwei definieren die linken und rechten Ränder, wobei der kleinere Wert den linken und der größere Wert den rechten Rand beschreibt. Die letzten beiden Parameter definieren die oberen und unteren Ränder; der kleinere den oberen und der größere den unteren Rand.
&1B	27	ESC		Kein Einfluß. Ignoriert.
&1C	28	FS	0...15 0...31 0...31	Farbenpaar für Farbstift setzen. Gleichwertig mit INK-Kommando. Der erste Parameter (MOD 16) definiert den Farbstift. Die beiden nächsten definieren die Farben (MOD 32).
&1D	29	GS	0...31 0...31	Farbenpaar für den nicht beschreibbaren Bildschirmrand. Gleichwertig mit BORDER-Kommando. Die zwei Parameter (MOD 32) definieren die beiden Farben.
1E	RS	RS		Cursor in die obere linke Ecke des Bildschirmteilbereichs setzen.
&1F	31	US	1...80	Cursor auf die angegebene Position setzen. Gleichwertig mit LOCATE-Kommando. Der erste Parameter definiert die Spalte und der zweite die Zeile.

9.2 Betriebssystem

Die Steuerung des CPC464 wird von einem anspruchsvollen Echtzeit-(real-time)-Betriebssystem übernommen. Das Betriebssystem steuert den 'Verkehrsfluß' durch den Computer, von der Eingabe bis zur Ausgabe.

Es stellt hauptsächlich das Bindeglied zwischen der Hardware und dem BASIC-Interpreter dar, z. B. bei der Funktion der wechselnden Schreibstiftfarben übergibt BASIC die Parameter und das Betriebssystem übernimmt die Weiterbehandlung: Ein Teil analysiert was zu tun ist, der andere überwacht den zeitlichen Ablauf.

Das Betriebssysem einer Maschine wird als sogenannte 'Firmware' bezeichnet und besteht aus Maschinencode-Routinen, die BASIC mit seinen Kommandos aufruft.

Während es über BASIC (außer mit dem Kommando ON BREAK GOSUB) fast unmöglich ist, den CPC464 zu blockieren, ist es relativ leicht, durch Zugriffe auf das Betriebssystem eine Situation zu erzeugen, die Sie nur durch vollständiges Zurücksetzen des Rechners wieder aufheben können. Dabei gehen natürlich das ganze Programm und alle Daten verloren!

Falls Sie versuchen sollten, mit POKE-Kommandos die internen Speicherinhalte zu verändern oder mit CALL-Aufrufen die Systemroutinen aufzurufen, sichern Sie vorher Ihr Programm, da es zerstört werden könnte.

Eine ausführliche Beschreibung der Betriebssystem-Firmware des CPC464 steht in einem zusätzlichen Handbuch, das weit über den Umfang dieser Einführung hinausgeht.

9.3 Unterbrechungen

Der CPC464 benützt die Unterbrechungsmöglichkeiten des Z80 in großem Maß, so daß ein Betriebssystem mit Multitasking-Funktionen (mehrere Anweisungen werden gleichzeitig bearbeitet) gegeben ist. Ein Beispiel dafür sind die AFTER- und EVERY-Kommandos, die in Kapitel 8 beschrieben sind. Die Priorität der Unterbrechungen hat folgende Reihenfolge:

Break [ESC] [ESC]
Zeitgeber 3
Zeitgeber 2 (und die drei Tonkanalwarteschlangen)
Zeitgeber 1
Zeitgeber 0

Unterbrechungen sollen erst dann eingebaut werden, wenn man die möglichen Zustände der verschiedenen Variablen zum Zeitpunkt der Unterbrechung gut durchdacht hat. Die Unterbrechungsbehandlungsroutine sollte ungewollte Veränderungen der Variablen des Hauptprogramms vermeiden.

Die Tonwarteschlangen haben unabhängige Unterbrechungen mit gleicher Priorität. Sobald eine solche Unterbrechung eingetreten ist, wird sie von keiner anderen nochmals unterbrochen, so daß in diesen Unterbrechungsbehandlungsroutinen anders als bei Zeitunterbrechungen die gleichen Variablen benutzt werden können.

Sind Unterbrechungen für eine Tonwarteschlange zulässig, so tritt sofort eine Unterbrechung ein, wenn die Tonwarteschlange für diesen Kanal nicht voll ist. Ansonsten tritt eine Unterbrechung nur ein, sobald der nächste Ton gestartet wurde und wieder Platz in der Warteschlange vorhanden ist. Sobald eine solche Unterbrechung eintritt, werden weitere tonabhängige Unterbrechungen gesperrt, so daß die Behandlungsroutine die Unterbrechungsmöglichkeit aktivieren muß, falls weitere Unterbrechungen gewünscht sind. Wird ein Ton ausgelöst oder der Zustand der Warteschlange getestet, werden ebenfalls weitere Unterbrechungen gesperrt.

Die Priorität der [ESC] [ESC]-Folge über allen anderen Unterbrechungen bewirkt, daß ein BASIC-Progrmm angehalten werden kann, ohne daß es verloren geht, solange nicht durch andere Anweisungen verhindert wurde, den Ablauf des Programms zu verändern.

9.4 CPC464 Assembler

Für größere Programme in Maschinensprache benötigt man einen Assembler. Der Assembler besteht aus einem nachladbaren Z80 Assembler mit Editierer, Disassembler und Steuerprogramm (Monitor).

10 Unterbrechungs-möglichkeiten

Themen dieses Kapitels:

* AFTER
* EVERY
* REMAIN
* Die zentrale Uhr

Falls Sie es noch nicht bemerkt haben sollten, eine der wichtigsten Neuheiten des CPC464 ist die Fähigkeit, Unterbrechungen in BASIC zu behandeln, d. h. **Amstrad** BASIC ist in der Lage, innerhalb eines Programms mehrere unterschiedliche Handlungen gleichzeitig auszuführen. Solch eine Fähigkeit wird manchmal 'Multitasking' genannt und kann mit den neuen Kommandos `AFTER` und `EVERY` angesprochen werden.

Diese Fähigkeit zeigt sich auch deutlich in der Behandlung von Tönen durch die Warteschlangen und Rendevouztechnik. Alle zeitabhängigen Ereignisse werden von der zentralen Uhr gesteuert, ein quarzgesteuerter Zeitgeber im Rechner. Sie überwacht den zeitlichen Ablauf und stimmt alle Handlungen im Rechner aufeinander ab – wie z. B. das Aufbauen des Bildes auf dem Bildschirm oder das Taktgeben im Prozessor. Wann immer eine Funktion der Hardware zeitabhängig ist, wird sie von der zentralen Uhr gesteuert.

Die Softwareseite ist mit den Kommandos `AFTER` und `EVERY` abgedeckt. Wie bei **Amstrad** BASIC üblich, sind diese Kommandos so benutzerfreundlich, daß sie auch das tun, was der Name verspricht. So wird z. B. nach `(AFTER)` der Zeit, die Sie in der Kommandozeile vorgegeben haben, in das gewünschte Unterprogramm gesprungen und dieses ausgeführt.

10.1 AFTER

Der CPC464 besitzt eine Echtzeituhr. Das `AFTER` Kommando ermöglicht, daß Unterprogramme erst nach einer geplanten Zeit ausgeführt werden können. Vier unabhängige Zeitgeber stehen zur Verfügung, die je ein eigenes Unterprogramm ansprechen können.

`AFTER` <Ganzzahliger Ausdruck>[,<Ganzzahliger Ausdruck>] `GOSUB` <Zeilennummer>.

Der erste <ganzzahlige Ausdruck> sagt aus, wie lange bis zur Ausführung des Unterprogramms gewartet werden soll. Die Zeit wird in Einheiten von 1/50 Sekunden gemessen.
Der zweite <ganzzahlige Ausdruck> gibt an, welcher der vier möglichen Zeitgeber verwendet werden soll. Der Wert muß zwischen 0 und 3 liegen, der Standardwert ist 0.

Sobald die vorgegebene Zeit abgelaufen ist, wird das Unterprogramm automatisch aufgerufen, gerade so, als ob ein `GOSUB`-Kommando an der Stelle im Programm angegeben wäre, an der es sich im Ablauf gerade befindet. Am Ende des Unterprogramms, das mit einem normalen `RETURN`-Kommando abgeschlossen wird, kehrt das Programm zu der Stelle zurück, wo es vorher unterbrochen wurde.

Die Zeitgeber haben verschiedene Unterbrechungsprioritäten. Der Zeitgeber 3 hat die höchste Priorität, der Zeitgeber 0 die niedrigste.

AFTER-Kommandos können zu jeder Zeit angegeben werden, wobei die zugehörigen Unterprogramme und Zeitgeber jeweils neu gesetzt werden. Die Zeitgeber sind dieselben, die auch im EVERY-Kommando verwendet werden, so daß ein AFTER-Kommando ein vorheriges EVERY-Kommando für einen bestimmten Zeitgeber überschreibt und umkehrt.

```
10 MODE 1:X=0
20 AFTER 45 GOSUB 100
30 AFTER 100,1 GOSUB 200
40 PRINT "AMSTRAD"
50 WHILE X<100
60 LOCATE #1,30,1:PRINT #1,X:X+1
70 WEND
80 END
100 PRINT "Peripherie"
110 RETURN
200 PRINT "und Software"
210 RETURN
```

Beachten Sie die Verwendung der zwei unterschiedlichen Ein-/Ausgabegeräte (Window), so daß das Hauptprogramm (Zeilen 50-80) und die Unterprogramme jeweils voneinander unabhängige Cursorpositionen für die Ausgabe benützen. Nachdem die Zeit abgelaufen ist, die Sie in der Kommandozeile (AFTER) vorgegeben haben, springt das Programm in das entsprechende Unterprogramm und führt dieses aus.

10.2 EVERY

Mit dem EVERY-Kommando kann ein BASIC-Programm in regelmäßigen Zeitabständen ein Unterprogramm aufrufen. Vier Zeitgeber stehen zur Verfügung, denen je ein Unterprogramm zugeordnet werden kann. Der Aufruf:

EVERY <ganzzahliger Ausdruck>[,<ganzzahliger Ausdruck>]
GOSUB<Zeilennummer>.

Der erste <ganzzahlige Ausdruck> sagt aus, wie lange zwischen jedem Unterprogrammaufruf gewartet werden soll. Die Zeit wird in Einheiten zu 1/50 Sekunden gemessen.

Der zweite <ganzzahlige Ausdruck> bestimmt, mit welchem der vier Zeitgeber gearbeitet werden soll. Die gültigen Werte sind 0 bis 3. Wird keine Zahl angegeben, so wird 0 angenommen. Sobald die vorgegebene Zeit abgelaufen ist, wird das Unterprogramm automatisch aufgerufen, gerade so, als ob ein GOSUB-Kommando an der Stelle im Programm angegeben wäre, an der es sich gerade befindet. Nach dem Ende des Unterprogramms, das mit einem normalen RETURN-Kommando abgeschlossen wird, läuft das Hauptprogramm an der Stelle weiter, wo es vorher unterbrochen wurde.

Die Zeitgeber haben unterschiedliche Unterbrechungsprioritäten. Der Zeitgeber 3 hat die höchste Priorität, der Zeitgeber 0 die niedrigste. Sobald der Zeitgeber abgelaufen ist, wird er zurückgesetzt und die Zeit bis zum nächsten Aufruf des Unterprogramms neu gewählt.

EVERY-Kommandos können zu jeder Zeit ausgegeben werden, wobei die zugehörigen Unterprogramme und Zeitgeber jeweils neu gesetzt werden. Die Zeitgeber sind dieselben, die auch im AFTER-Kommando verwendet werden, so daß ein EVERY-Kommando ein vorheriges AFTER-Kommando für einen bestimmten Zeitgeber überschreibt und umgekehrt.

```
10 MODE 1:X=0
20 P100=0:EVERY 10 GOSUB 100
30 P200=0:EVERY 12,1 GOSUB 200
40 PRINT "AMSTRAD"
50 WHILE X<200
60 LOCATE #1,30,1:PRINT #1,X:X=X+1
70 WEND
80 LOCATE 1,20:END
100 DI:PEN P100:LOCATE 1,2:PRINT "Peripherie":EI
105 IF P100=0 THEN P100=1 ELSE P100=0
110 RETURN
200 PEN P200:LOCATE 1,3:PRINT "und Software"
205 IF P200=2 THEN P200=3 ELSE P200=2
210 RETURN
```

Beachten Sie die Verwendung der DI- und EI-Kommandos, womit die Zeitunterbrechungen verboten bzw. wieder erlaubt werden, während die dazwischen liegenden Kommandos ausgeführt werden. Dies bewirkt, daß der Zeitgeber 1 mit der höheren Priorität niemals den Programmablauf unterbrechen kann, während die Unterbrechungsbehandlung für den Zeitgeber 0 (Zeilen 100-110) abläuft und dadurch die PEN- und LOCATE-Zuordnungen vor dem PRINT-Kommando umsetzen kann.

10.3 REMAIN

Diese Funktion gibt die noch verbleibende Zeit eines der vier Zeitgeber zurück. 0 wird zurückgegeben, wenn der Zeitgeber nicht eingeschaltet ist. Außerdem schaltet die Funktion den entsprechenden Zeitgeber ab. Die Funktion wird folgendermaßen aufgerufen:

REMAIN (<Ganzzahliger Ausdruck>)

ANHANG 1

Die Kunst des Machbaren

Als Anfänger in der elektronischen Datenverarbeitung wird man oft versuchen, sich anhand einiger Vergleichskriterien über die Fähigkeiten und Grenzen eines Computers klarzuwerden. Deshalb soll in diesem Abschnitt umfassend erklärt werden, was und warum etwas passiert.

Abschießen!

Möglicherweise war für Sie der einzige Anlaß diesen Computer zu kaufen der Reiz der vielen ausgeklügelten Computerspiele, die auf dieser ‚Hardware' ablaufen können. Trotzdem wird es Sie vielleicht interessieren, wieviel verschiedene Dinge unter dem Oberbegriff ‚Hardware' zusammengefaßt werden:

Hardware sind zunächst einmal alle Dinge, die Sie unter den Arm nehmen und davontragen können, wie etwa das Grundgerät mit der Eingabetastatur, der Monitor, die Verbindungskabel und so weiter. Hardware ist alles, was nicht speziell als Software angesprochen wird, nämlich Programme, Handbücher und auf Cassette gespeicherte Informationen.

Einige charakteristische Eigenschaften eines Computers basieren auf seiner Hardware – zum Beispiel die Fähigkeit zur Farbdarstellung auf dem Monitor oder einem Farbfernsehgerät. Aber dann ist die Software dran, um die Möglichkeiten der Hardware zu nutzen und die gewünschten Zeichen und Figuren auf dem Bildschirm sichtbar zu machen.

Die Hardware selbst ist für die Steuerung des Elektronenstrahls auf der Leuchtschrift im Inneren der Bildröhre des Monitors verantwortlich – die Software bringt Sinn und Ordnung in diesen Vorgang, indem sie der Hardware sagt, wann und wie sie etwas durchführen muß. Von ihr gehen die Zeitabstimmung, genaue Kontrolle und die Festlegung der Funktionsabfolge aus, um auf dem Bildschirm ein abhebendes Raumschiff oder auch einfach nur einen Buchstaben erscheinen zu lassen, wenn Sie auf die richtige Taste drücken.

Frage: Was macht nun den einen Computer besser als einen anderen?

Hard- und Software sind für sich allein genommen vollkommen nutzlos, einzeln kann man mit ihnen überhaupt nichts anfangen. Erst die Verbindung von Hardware und Software macht es dem Computer möglich, vielseitige Aufgaben zu erfüllen. Nun kann man anhand einiger Grundüberlegungen die Qualität sowohl von Hard- als auch Software einstufen.

Für Personalcomputer gelten heute allgemein folgende Kriterien:

1. Die Bildschirmauflösung – die kleinsten unterscheidbaren Bildelemente

Die auf dem Bildschirm erreichbare Auflösung beruht auf einer Kombination verschiedener Faktoren, wie der Anzahl von Farben, die dem Programmierer zur Verfügung stehen, die Anzahl der einzeln ansteuerbaren Bildpunkte (in der Fachsprache: Pixel) und auch die Zahl der Textzeichen.

Bei einem Vergleich mit irgendeinem Computer dieser Preisklasse werden Sie feststellen, daß Ihr CPC464 in allen diesen Punkten sehr gut abschneidet.

2. Der BASIC-Interpreter

Fast in jedem Heimcomputer ist ein BASIC-Interpreter eingebaut, der es dem Anwender erlaubt, mit relativ geringem Aufwand eigene Programme zu entwickeln. Die Programmiersprache BASIC, die auch in Ihrem Gerät eingebaut ist, ist selbst ein gewaltiges, kompliziertes Programm. Seit ihrer ‚Erfindung' in den USA wurde es durch die Erfahrung von mehr als einer Million Mannjahren weiterentwickelt. Weltweit ist der ‚Beginners All Purpose Symbolic Instruction Code' die am weitesten verbreitete Computersprache, und wie für jede Sprache, so gibt es auch für BASIC etliche Dialekte.

Die im CPC464 verwendete Version ist einer der weitgehend kompatiblen BASIC-Dialekte. Gewöhnliche BASIC-Programme, die für das Disketten-Betriebssystem CP/M geschrieben wurden, können in den meisten Fällen ohne Änderung ablaufen. Darüber hinaus ist es ein sehr schnelles BASIC – d. h. die Berechnungen werden sehr schnell durchgeführt. Wahrscheinlich wird es Sie zunächst kaum beeindrucken, ob ein Computer 0,05 s oder 0,07 s braucht um 3 mit 5 zu multiplizieren und das Ergebnis anzuzeigen, aber in einem Graphik-Programm mit vielen tausenden solch einfacher Rechenoperationen kann diese geringfügige Differenz am Ende einen großen Unterschied für die Leistungsfähigkeit eines Systems bedeuten.

In Zukunft werden Sie vermutlich oft den Ausdruck ‚Maschinensprache' hören: Dies ist die Form von Anweisungen, die vom Prozessor direkt verarbeitet werden kann. In dieser Maschinensprache braucht der Rechner wesentlich weniger Zeit um die gestellte Aufgabe zu lösen, liefert das Ergebnis 5 bis 15 Mal so schnell als wenn die Bearbeitung des Problems über den BASIC-Interpreter läuft.

Aber es kann 5 bis 50 Mal so lange dauern ein entsprechendes Programm in Maschinensprache zu schreiben, als wenn Sie es in BASIC erstellen.

Das BASIC in Ihrem **Amstrad** Computer gehört zu den schnellsten und umfangreichsten, die heute in irgendeinem Heimcomputer zu finden sind.

Es hat viele Eigenschaften, mit deren Hilfe der erfahrene BASIC-Programmierer die Starrheit einer höheren Programmiersprache durchbrechen kann, um erstaunliche Bild- und Toneffekte zu erzeugen.

3. Ausbaufähigkeit

Bei den meisten Computern wird viel Wert auf die Möglichkeit gelegt, zusätzliche Hardware wie z. B. Drucker, Joysticks und Floppy-Disk Laufwerke anschließen zu können. Paradoxerweise erfordern einige der erfolgreichsten Heimcomputer erst den Anbau zusätzlicher Geräte die unter dem Namen ‚Erweiterungsschnittstellen' bekannt sind, bevor auch nur irgendein einfacher Drucker oder Joystick angschlossen werden kann.

Nicht immer weiß der Käufer im voraus, was er in Zukunft alles brauchen wird, sonst wäre für ihn auf lange Sicht eine Maschine billiger, die von Haus aus einen Parallel-Drucker (Centronics-kompatibel) und Joysticks für seine Spiele bedienen kann.

Der CPC464 hat einen eingebauten Anschluß für Centronics-Drucker, Anschlußmöglichkeit für bis zu zwei Joysticks, einen Stereo-Ausgang und einen umfangreichen Erweiterungsbus, an den man Controller für Disketten-Laufwerke, zusätzliche Erweiterungs-ROMs, serielle (RS232) Schnittstellen, usw. anschließen kann.

Ein ROM (Read Only Memory – Nur lesbarer Speicher) ist eine integrierte Schaltung, die auf Dauer gespeicherte Programminformationen enthält. Das in Ihrem Computer mitgelieferte BASIC ist in einem solchen ROM gespeichert. Zusätzlich erhältliche Programme können entweder das eingebaute ROM ‚erweitern', oder seine Funktion vollkommen ersetzen.

Geräte für TV-Spiele benutzen Software in Steck-Modulen. Ein Steck-Modul ist eigentlich ein ROM, das in einem hübschen Plastikgehäuse mit passendem Stecker geliefert wird, um es leicht ein- und ausstecken zu können. Somit bietet ein ROM die gleiche Funktion wie eine Magnetband-Cassette, nämlich Programm-Informationen zu liefern.

Während der Computer jedoch die Informationen aus dem ROM quasi ohne Zeitverzug laden kann, braucht er zum Laden größerer Programme von einer Cassette mehrere Minuten. Der Hauptvorteil des ROM's liegt also in der größeren Bequemlichkeit.

Man kann ein ROM nicht dazu verwenden, um Informationen zu speichern, sie herausnehmen und in einem anderen Computer einzuspeichern, wie man das mit einer Magnetband-Cassette machen kann.

Mit der Ausbaufähigkeit ist sichergestellt, daß Ihr Computer an die meisten zukünftigen Entwicklungen bezüglich Software und Zusatzhardware Anschluß findet. Das CPC464 System hat eine komplette und vollständig dokumentierte Ausbaumöglichkeit.

4. Geräusche und Musik

Von den Möglichkeiten zur Tonerzeugung hängt es ab, ob ein Computer wie eine scheppernde Gießkanne klingt – oder ob man ihn als ernstzunehmendes Instrument für elektronische Musik verwenden kann.

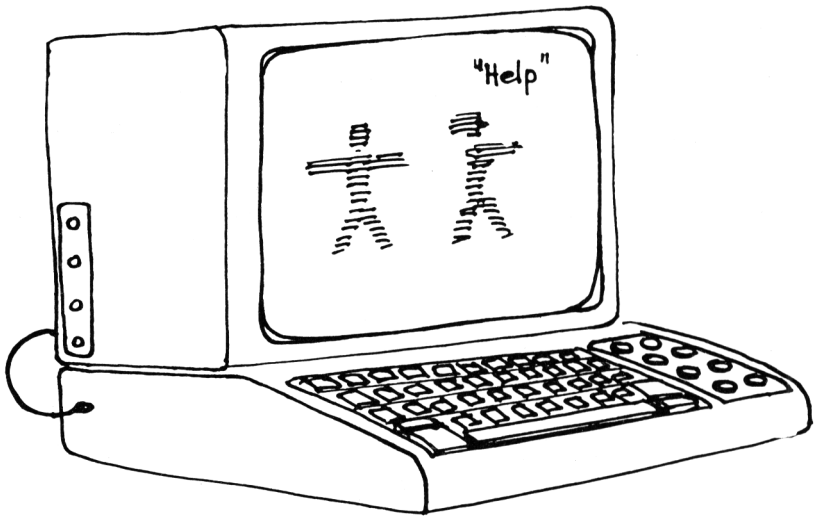
Der CPC464 benutzt einen Tongenerator mit 3 Kanälen, der Tonumfang beträgt 7 Oktaven. Über die Beeinflussung von Amplitude und Hüllkurve wird eine äußerst ansprechende Musikqualität erreicht. Das 3-Kanal-System ordnet einen Kanal der linken, den zweiten der rechten Seite zu, der dritte Kanal scheint akustisch aus der Mitte zu kommen. Mit diesen Möglichkeiten können ausgefeilte Geräuscheffekte erzielt werden, ja sogar Kulisseneffekte, die simultan zur Bewegung am Bildschirm verlaufen.

Letztendlich müssen Sie als Benutzer selbst entscheiden, welche der Eigenschaften Ihres Computers für Sie am wichtigsten ist. Wir können nur hoffen, daß Sie sie alle ausprobieren, damit Sie das Beste aus Ihrem Gerät herausholen.

Warum ist es nicht möglich, daß ...

Von Computerneulingen wird oft die Frage gestellt, warum ein so weit entwickeltes Gerät wie der CPC464, auf dem neuesten Stand der Technik, nicht in der Lage ist, die Art von Bildern auf den Bildschirm zu bringen, die dem Standard eines jeden Fernsehgerätes entsprechen.

Frage: Warum, z. B., kann ein Computer kein Bild mit einer natürlich über den Bildschirm laufenden Person darstellen – warum sehen alle Figuren aus wie Strichfiguren mit ‚Charlie Chaplin' Bewegungen?



Die Antwort ist sowohl einfach wie komplex. Die einfache Antwort ist, daß Sie nicht dazu verleitet werden sollen zu glauben, daß der Bildschirm so raffiniert ist wie der Schirm Ihres Fernsehgerätes. Ein Fernseher arbeitet mit linearen Informationen, die scheinbar eine unendlich feine Abstufung (Auflösung) zwischen hell und dunkel, sowie über alle Nuancen des Farbspektrums ermöglichen. Das bedeutet, daß so ein Bildschirm rund 20 Mal soviel Speicherkapazität benötigt, wie umgerechnet ein Heimcomputer für seine Videoausgabe braucht.

Das ist nur ein Teil des Problems, da solche Bildbewegungen verlangen, daß diese enorme Speicherkapazität sehr schnell verarbeitet werden muß (ca. 50 Mal pro Sekunde). Das erreichen nur Maschinen, die einige tausend Mal teurer sind wie ein Heimcomputer – wenigsten in der nahen Zukunft.

Bis die Kosten für Hochgeschwindigkeitsspeicher drastisch fallen, und das werden sie irgendwann, müssen Kleincomputer mit vergleichbar wenig Speicher auskommen, um den Bildschirm zu steuern. Das bringt niedrigere Auflösung und stotternde Bewegungen. Überlegte Hardwareplanung und gute Programmierung können viel dazu beitragen, diese Situation zu mildern, aber wir sind noch sehr weit davon entfernt, auf billigen Computern realistische Bewegungen und lebendige Bilder darzustellen, wenigstens so wie ein Zeichentrickfilm.

Frage: Warum kann ich nicht zum Computer gehen und eine Seite mit einfachem Text direkt in die Maschine eintippen?

Lassen Sie sich nicht dadurch verleiten, daß der Computer wie eine Schreibmaschine mit einer elektronischen Anzeige aussieht. Der Bildschirm ist kein Stück elektronisches Papier – d. h., er gibt Ihnen einfach nur die Möglichkeit, mit der Programmiersprache (und den Programmen) im Speicher der Maschine Kontakt aufzunehmen.

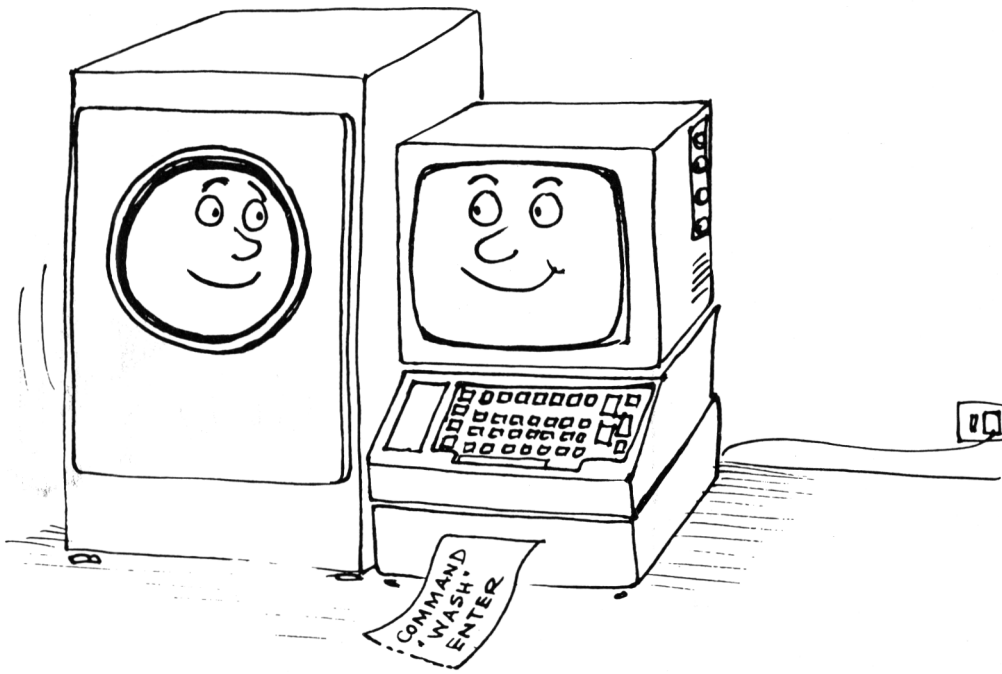
Solange Sie ihm nichts anderes sagen, versucht der Computer sämtliche Zeichen, die Sie über die Tastatur eingeben, als Programmanweisungen zu interpretieren. Wenn Sie auf [ENTER] drücken, schaut sich der Computer das an, was Sie eingetippt haben, um zu sehen, ob es einen Sinn für sein eingebautes BASIC ergibt. Wenn nicht, reagiert der Computer ganz stur mit der Meldung:

```
SYNTAX ERROR
```

Wenn Sie aber gerade ein Textverarbeitungsprogramm aktiv im Speicher haben, dürfen Sie beliebig Wörter eintippen, auf [ENTER] drücken, und weitermachen, als ob Sie auf einem elektronischen Stück Papier in einer elektronischen Schreibmaschine arbeiten würden.

Aber dazu müssen Sie erst ein Textverarbeitungsprogramm in den Computerspeicher von einer Datencassette geladen haben.

Es ist ‚als ob' ein Computer alle die Maschinen vereint, die zu Hause und im Büro bekannt sind – der fernseherartige Bildschirm, die Tastatur, der Cassettenrecorder – aber diese Ähnlichkeiten sind nur oberflächlich. Der Computer ist eine Kombination vertraut aussehender Hardware mit einer eigenen Persönlichkeit.



BEGRIFFS-ERKLÄRUNGEN

Einige häufig verwendete Begriffe aus der Welt der Datenverarbeitung und deren Erklärung für die Benutzer des CPC464 ...

Abenteuerspiele (Adventure Game)

Für einige das Höchste, für andere schlicht langweilig. Ein Computerspiel mit viel Text, bei dem der Spieler eingeladen ist, an einer Reihe von Zufallsereignissen teilzunehmen und sich in einem Irrgarten oder Labyrinth zurechtzufinden.

Abschneiden (Truncated)

Eine Zahl oder ein Text werden gekürzt, indem die führenden oder hintenstehenden Zeichen entfernt werden. Diese Funktion kann mit Absicht benutzt werden bei der Rundung einer Zahl. Unabsichtlich angestoßen gehen die zusätzlichen Zeichen verloren, da für die Zahl oder Zeichenkette nicht mehr Platz zur Verfügung steht.

Akkumulator (Accumulator)

Eine Speicherstelle im Microprozessor-Schaltkreis, im Herz des Microcomputers, in der Daten während ihrer Verarbeitung vorübergehend zwischengespeichert werden. Er wird ständig in der Maschinencode-Programmierung verwendet – BASIC Anwender interessiert es nicht, daß es ihn gibt!

Akkustikkoppler (Acoustic coupler)

Auch bekannt als akustisches Modem. Eine elektronische Zusatzeinrichtung, die einen Telefonhörer mit einem Computer verbindet und ihn damit an das normale Telefonnetz anschließt. Auf diese Weise kann ein Computer mit öffentlichen Informationssystemen wie z. B. Btx verkehren, oder mit anderen Benutzern von Heimcomputern usw., um Software auszutauschen oder nur Daten und Informationen zu empfangen.

Algorithmus (Algorithm)

Ein großartig klingender Name für eine komplizierte Formel oder Rechenaufgabe. Eine Folge logischer und arithmetischer Schritte um eine genau definierte Aufgabe mit dem Computer zu erledigen.

Adresse (Address)

Die Zahl in einer Anweisung, die die Lage einer 'Zelle' im Speicher des Computers identifiziert. Über ihre Adresse kann eine bestimmte Speicherstelle gefunden und ihr Inhalt 'gelesen' werden. Wenn es sich um ein RAM handelt, kann sowohl gelesen als auch – nach einer Änderung – zurückgeschrieben werden.

Alphanumerisch (Alphanumeric)

Eigenschaft, die die Zeichenmenge aus Buchstaben und Ziffern und Sonderzeichen (auch graphischen Symbolen) bezeichnet.

ALU

Abkürzung für Arithmetic Logic Unit (arithmetisch-logische-Einheit). Der Teil des Mikroprozessors, der die arithmetischen und logischen Operationen ausführt. Nur wichtig.beim Programmieren im Maschinencode.

A/D Analog

Ein Zustand, bei dem Veränderungen zwischen einem Anfangs- und einem Endpunkt stufenlos sind, und nicht in einzelnen Schritten. Computer arbeiten digital, so daß die in der natürlichen Welt meist analog vorkommenden Ereignisse erst in digitale Einheiten umgeformt werden müssen, bevor ein Computer etwas damit anfangen kann.

Animation

Zeichentrickfilme sind die bekannteste Art von Animation. Bei der Computer Animation werden graphische Darstellungen bewegt, so daß ein Gefühl von 'echter' Bewegung vermittelt wird.

Anweisung (Statement)

Ein Befehl oder eine Folge von Befehlen in einem Programm.

Anwendungsprogramm (Applications program)

Ein Programm für eine spezielle Aufgabe, im Gegensatz zu einem allgemein anwendbaren Software-'Tool', wie z. B. Assembler, Sortierprogramm, Drucksteuerprogramm, usw.

Architecture

Allgemeiner Aufbau und interne Beziehung zwischen Datenbus, peripheren Geräten und CPU eines Mikrocomputers. Kein Thema für Leser dieser Begriffserklärungen – noch nicht!

Argument

Eine unabhängige Variable, z. B. in dem Ausdruck x+y=z, sind x und y Argumente.

Array

Eine zweidimensionale Matrix (Gitter), in der Daten gespeichert und mit 'waagrechten' und 'senkrechten' Koordinaten adressiert werden.

ASCII

American Standard Code for Information Interchange. Eine weit verbreitete Art der Darstellung von Nummern, Buchstaben und sonstigen Zeichen, die von der Tastatur eingegeben werden können, bzw. durch andere Kommandos erstellt werden. Die Codes des CPC464 sind in Anhang III aufgeführt.

Assembler

Maschinennahe Programmiersprache mit symbolischer Schreibweise für Befehle.

Auflösung (Resolution)

Die Möglichkeit festzustellen, wo ein Element einer Bildschirmausgabe endet und wo das nächste beginnt. Manchmal auch für die Fähigkeit des Computers, Errechnungen mit sehr großen Zahlen durchzuführen.

Ausdruck (Expression)

Eine einfache oder komplizierte Formel, im Programm zur Berechnung von Daten – die Formel beschreibt meist die Art von Daten, die bearbeitet werden kann.

Ausgabe (Output)

Alles was vom Computer als Ergebnis einer Errechnung kommt.

Austesten (Debugging)

Fehlersuche im Programm durch eine Kombination aus Versuchen und wissenschaftlichen Methoden.

Automatenspiel (Arcade Game)

Computerspiele besonderer Art mit viel 'Aktion', in denen Eindringlinge aus dem Weltall angreifen und besiegt werden, hungrige Monster-Figuren durch Irrgärten jagen und auffressen und der Spieler seine Figuren steuern muß, daß sie dem unangenehmen 'Tod' entgehen. Sie machen Spaß und trainieren die Reflexe, bieten aber nur wenig für den, der sich ernsthaft mit dem Computer beschäftigt.

BASIC

Beginners' All-purpose Symbolic Instruction Code (ein allgemeiner symbolischer Befehlscode für Anfänger). Eine interpretative Programmiersprache, die in fast allen Heimcomputern benutzt wird. BASIC ist leicht zu erlernen und einfach zu benutzen, da Programme während ihrer Entwicklung zusammengestückelt und getestet werden können, und nicht wie bei Compilern gleich das komplette Programm vorliegen muß.

Basis (Base)

Basis im Zahlensystem. Das Binärsystem hat Basis 2, das Dezimalsystem hat die Basis 10 und das Hexadezimale (sedezimale) System hat die Basis 16. Siehe Anhang II für weitere Informationen.

Baud

Ein Bit pro Sekunde. Die Maßeinheit für die Übertragungsgeschwindigkeit bei der Datenübertragung.

BCD

Binär codiertes Dezimalsystem. Ein System zur Codierung von Dezimalzahlen, in dem jede Zahl mit vier binären Ziffern dargestellt wird.

Befehl (Instruction)

Eine Anforderung oder ein Kommando an den Computer, eine bestimmte Operation auszuführen. Eine Sammlung oder Folge von Anweisungen ergeben zusammen ein Programm.
Die primären logischen und mathematischen Prozesse, die ein Microprozessor durchführt. Jeder Befehl einer höheren Programmiersprache (auch Assemblerbefehle) müssen in einzelne Befehle aufgegliedert werden, die von einer Computer-CPU verstanden werden können. Ein einziges Kommando kann u. U. viele Befehle der CPU auslösen.

Bemerkung (Remark)

Eine nicht ausführbare Programmanweisung (REM), im Programm zu dokumentieren, was an dieser Stelle getan wird oder auch zur Kennzeichnung des Programmentwicklungsstandes.

Benutzerdefinierte Taste (User defined key – UDK)

Der CPC464 hat 32 Tasten, die redefiniert werden können, um verschiedene Aufgaben zu erledigen. Jede dieser Tasten kann bis zu 32 Zeichen aufnehmen.

Betriebssystem (Operating system)

Software, welche die Verwaltung des Speichers und die Koordinierung der Aktionen des Computers steuert.

Bildschirmeditor (Screen Editor)

Ein Aufbereitungsprogramm für Texte oder Programme, bei dem der Cursor an jede Stelle des Bildschirms positioniert werden kann, um die dort stehenden Zeichen zu ändern.

Binär (Binary)

(Siehe Basis) Das Zahlensystem mit der Basis 2, in dem alle Zahlen nur mit den beiden Ziffern 0 und 1 dargestellt werden.

Binärzahl (Binary number)

Eine Zahl in binärer Schreibweise. Bei der Programmierung des CPC464 werden sie mit einem vorangestellten &X angegeben (z. B. bedeutet &X0101 in dezimaler Schreibweise 5).

Bit

Kurzform von Binary digIT (Binärziffer). Eine Binärziffer kann im binären Zahlensystem nur den Wert 0 oder 1 annehmen.

Bitsignifikant

Jedem der acht Bits eines Bytes ist eine gesonderte Bedeutung zugeordnet. Zur Auswertung muß jedes Bit getestet werden.

Boolesche Algebra (Boolean algebra)

Logische Verknüpfung von Operanden, die nur zwei Ergebnisarten (wahr, unwahr) kennt, die üblicherweise mit 0,1 dargestellt werden.

Bus

Eine Gruppe von Verbindungen entweder im Computer selbst oder als Verbindung zu Peripheriegeräten, womit Informationen über den Zustand der CPU, der RAM oder andere Hardwareeigenschaften übertragen werden. Der CPC464 Bus ist auf der größeren der beiden Steckerleisten auf der Computerrückseite ansprechbar.

Byte

Eine Gruppe von acht Bits, welche die kleinste adressierbare Einheit im Speicher ist. Siehe Anhang II für weitere Informationen.

CAD

Computer Aided Design (Computer Unterstützter Entwurf). Normalerweise ein Zusammenspiel von Rechenleistung und Computergraphik als elektronisches Reißbrett, obwohl jede Art von Errechnungen für 'Entwürfe' als CAD gelten können.

CAE

Computer Aided Education (Computer unterstützte Schulung). Noch ein Schlagwort der Computerwelt. Hiermit ist das ganze Thema der Schulbildung mit Hilfe von Computern gemeint. CAI (Computer unterstützter Unterricht) und CAL (Computer unterstütztes Lernen) sind zwei Aspekte von CAE.

Cassette

Außer dem bekannten Magnetband versteht man hierunter auch alles sonst, was irgendwie 'verpackt' kommt, ROM-Software usw.

Chip

Ein irreführender, jedoch umgangssprachlicher Begriff für eine integrierte Schaltung. Der Chip ist in Wirklichkeit eine dünne Scheibe aus speziell hergestelltem Silizium, auf der die Schaltung untergebracht ist.

Code

Außer in seiner normalen Bedeutung wird dieses Wort von Programmierern oft als Abkürzung von 'Maschinencode' benutzt.

Compiler

Ein komplexes Programm, das andere in einer höheren Programmiersprache als BASIC geschriebene Programme in das Befehlsformat des Microprozessors umwandelt. Dadurch wird eine höhere Ablaufgeschwindigkeit ermöglicht.

Computer der fünften Generation (Fifth generation computers)

Meist sehr große Computer, die unter Anwendung sogenannter künstlicher Intelligenz selbst programmiert werden können.

Computergenerationen (Computer generations)

Klassifizierung der großen technischen Meilensteine der Computerentwicklung. Die Gruppierungen, die sich daraus ergeben, werden als Generationen bezeichnet.

CP/M

Der de facto Standard von Digital Research für 8-Bit-Computer mit Plattenbetriebssystemen.

CPU

Central Processing Unit (Zentrale Prozessoreinheit). Das Herz eines jeden Computersystems, in dem Anweisungen interpretiert und deren Ausführung ausgelöst werden. In einem Microcomputer ist die CPU der Microprozessor selbst.

Cursor

Ein beweglicher Zeiger, der anzeigt, wo das nächste Zeichen auf dem Bildschirm erscheint.

Cursor Kontrolltasten (Cursor control keys)

Tasten, die den Cursor auf dem Bildschirm bewegen und auf denen Pfeile abgebildet sind. Sie werden oft zur Kontrolle von Bewegungen in Automatenspielen benutzt.

Datei (File)

Eine Datensammlung, die meistens auf Cassette oder Platte gespeichert ist, obwohl manche Computer auch RAM-Dateien unterstützen.

Datenaufnahme (Data capture)

Die Art, wie Daten von einem externen Gerät aufgenommen werden, das in beliebiger Weise an den Computer angeschlossen ist.

Datenfluß (Stream)

Der Weg, der bei der Ausgabe vom Computer benutzt wird, z. B. der Bildschirm, die Cassette oder der Drucker.

Datenbasis (Database)

Ein Bereich mit beliebigen Datenarten in verschiedenen adressierbaren Formaten.

Datenübertragung (Download)

Die Übertragung von Information von einem Computer zum anderen, wobei der Empfänger als 'downloading' Maschine bezeichnet wird. Der andere Computer ist dann die 'uploading' Maschine.

Dezimale Schreibweise (Decimal system)

Auch bekannt als Dezimalsystem, ein Zahlensystem zur Basis 10 mit Ziffern von 0 bis 9, mit dem Einser, Zehner, Hunderter, Tausender usw. dargestellt werden.

Diagnostik (Diagnostic)

Die Meldung, die vom Computer automatisch ausgegeben wird, um mit der dazu gehörenden Erklärung auf einen Fehler im Programm hinzuweisen.

Digital

Die Art, Gesamtabläufe in Einzelschritte aufgelöst darzustellen. Das Gegenteil von Analog.

Dienstprogramm (Utility)

Jedes komplette Programm für eine oft gebrauchte Standardfunktion, wie z. B. Daten sortieren, oder Dateien kopieren.

Digitiser

Ein Gerät, um Daten durch Abtasten in den Computer einzugeben. Wird sehr oft auch im Zusammenhang mit Graphiktabletts verwendet.

Diskette (Floppy disk)

Eine herausnehmbare Magnetplatte mit 5,25 oder 8 Zoll Durchmesser zur Speicherung von Daten. Die dünne Scheibe ist in einer quadratischen Hülle untergebracht. Sie kann erheblich mehr Daten aufnehmen als eine Cassette, ist viel schneller und kostet mehr.

Dokumentation (Documentation)

Die Handbücher, die mit Computern oder Software ausgeliefert werden und erklären, wie sie benutzt werden.

DOS

Plattenbetriebssystem (Disk Operating System). Die Software, die alle Funktionen einer Magnetplatte steuert, ähnlich einem Parkplatzwächter, der die Autos (Daten) auf ihren Platz (logische Plattenaufteilung) einweist.

E/A (I/O)

Ein-/Ausgabe (Input/Output)

Echtzeit (Real time)

Ereignisse, die sich vor Ihren Augen abspielen, im Gegensatz zu jenen, die erst nach Ablauf eines Prozesses zu Tage treten, in dem sie erstellt werden.

Editieren (Editing)

Das Korrigieren oder Ändern von Daten, einem Programm oder Texten.

Editor

Ein Programm, normalerweise im ROM, welches das Editieren ermöglicht.

Eingabe (Input)

Alles was in den Computerspeicher gelangt, über die Tastatur, den Cassetten-Datenrecorder, das Plattengerät, eine serielle Schnittstelle oder eine andere Eingabequelle.

EPROM

Lösch- und programmierbarer Speicher, der nur gelesen werden kann. Ähnlich einem PROM, nur sind die Daten darauf mit ultraviolettem Licht löschbar, so daß neue Programme darauf gespeichert werden können. Ein EEPROM ist ähnlich, wird aber elektrisch gelöscht.

Fehler (Bug)

Sie können ganz verschieden auftreten, als unerwarteter Programmablauf verursacht durch eine falsche Anwendung des Programms, (z. B. ändert sich die Bildschirmfarbe, wenn gleichzeitig vier Tasten gedrückt werden), oder als Kette von Fehlern, die komplett und unwiderruflich ein Programm abstürzen läßt und sämtliche Daten im Speicher löscht.

Festkommazahl (Fixed-point number)

Eine Zahl, deren Dezimalpunkt an einer fest definierten Position steht.

Firmware

Software im ROM – ein Bindeglied zwischen reiner Software und reiner Hardware.

Flußdiagramm (Flowchart)

Eine systematische Darstellung von Programmablauf und logischen Prozessen, womit die Reihenfolge der Ereignisse während der Ausführung des Programms verfolgt werden kann.

Forth

Eine sehr schnelle Programmiersprache, mit Geschwindigkeit und Komplexität zwischen einer höheren Sprache und Maschinencode. Nicht für Anfänger geeignet.

Funktionstaste (Fuction key)

Eine Taste der Tastatur, der eine besondere Funktion zugeordnet ist, die zusätzlich oder an Stelle der eigentlich zugeordneten Funktion dieser Taste verwendet werden kann. Der CPC464 hat einige Tasten, die als Funktionstasten definiert werden können, wobei ein einzelner Tastendruck bis zu 32 Textzeichen auslösen kann, z. B. oft benutzte Anweisungen oder Befehle für Peripheriegeräte wie Drucker oder Modems.

Ganzzahl (Integer number)

Eine Zahl ohne Bruchteil, d. h. ohne Ziffern rechts vom Dezimalpunkt. Dagegen ist eine reelle Zahl eine Ganzzahl zuzüglich dem Bruchteil.

Gatter (Gate)

Logische Gatter erlauben die Weitergabe von Daten, falls bestimmte Bedingungen erfüllt sind. Es gibt viele Arten (OR, AND, XOR usw.). Siehe auch Boolesche Algebra.

Geräusch (Noise)

Der Tongenerator des CPC464 ermöglicht dem Anwender mit dem SOUND-Kommando Geräuscheffekte zu erzeugen, wie z. B. Explosionen.

Gleitkommazahl (Floating-point number)

Eine reelle Zahl, deren Dezimalpunkt an einer gewünschten Position steht. Diese Methode bietet sich für die Darstellung sehr großer Zahlen an.

Graphik (Graphics)

Der Teil von der Bildschirmausgabe, der nicht unmittelbar mit der Textzeichenausgabe zu tun hat, z. B. das Zeichnen von Linien, Kreisen usw. Mit einem geeigneten Drucker kann auch eine Ausgabe auf Papier erfolgen.

Graphikcursor (graphics cursor)

Ähnlich dem Textcursor, jedoch für den Graphikbildschirm. Zwar ist er beim CPC464 unsichtbar, aber notwendig für das Zeichnen von Graphiken. Nicht mit graphischen Zeichen verwechseln (Anlage III), die ein Teil des Zeichenvorrats sind und beim Textcursor ausgegeben werden.

Graphikmodus (Graphics mode)

Frühere Microcomputer mußten in den jeweiligen Modus zur Verarbeitung von entweder Text oder Graphik umgeschaltet werden. Die modernen Personal Computer können beide Funktionen gleichzeitig behandeln.

Graphiktablett (Graphics tablett)

Ein Eingabegerät, das die Koordinatenpunkte eines Bildes oder einer Zeichnung zur Verarbeitung an den Computer weitergibt. Eine Art Analog/Digitalwandler.

Graphisches Zeichen (Graphics charakter)

Besondere Zeichen oder Formen, die für die Erstellung von Bildern sehr nützlich sind. Der CPC464 hat einen vollständigen Satz davon, dieser ist in Anlage III beschrieben.

Halbbyte (Nibble)

Ein vier Bit großer binärer Ausdruck. Jede der hexadezimalen Ziffern im Ausdruck &F6 stellt ein Halbbyte dar.

Hard Copy

Papierausdruck von einem Programm oder sonstigen Texten – oder von Graphiken. Der vorübergehende Ausdruck am Bildschirm heißt Softcopy.

Hardware

Die elektronischen und mechanischen Teile eines Computersystems, alles das weder Software noch Firmware ist.

Hexadezimaldarstellung (hexadecimal notation)

Zahlen zur Basis 16, siehe Anhang II. Im CPC464 durch ein vorgestelltes & bzw. &H gekennzeichnet.

Höchstwertiges Bit (MSB – Most Significant Bit)

Das in einer Binärzahl am weitesten links stehende Bit.

Höhere Programmiersprache (Higher level language)

Programmiersprache, die der normalen Sprechweise ziemlich nahe kommt, so daß sie sehr leicht interpretiert werden kann. Langsamer als Programme in Maschinencode, jedoch dafür viel einfacher zu verstehen, wie z. B. BASIC.

IEEE-488

Eine der Standardschnittstellen für den Anschluß von Geräten an einen Microcomputer. Ähnlich der Centronics Parallelschnittstelle, jedoch nicht völlig kompatibel.

Informationstechnik (Information technology)

Alles was mit der Anwendung von Elektronik bei Nachrichtenverarbeitung und Kommunikationen zu tun hat: Textverarbeitung, Datenfernverarbeitung, PRESTEL, usw.

Initialisieren (Initialise)

Ein System einschalten, oder Vorbelegung von Variablen vor dem eigentlichen Hauptprogramm – z. B. Variablen als Ganzzahlen zu deklarieren.

Integrierte Schaltung (Integrated circuit)

Eine Sammlung winzig kleiner elektronischer Schaltungseinheiten auf einem einzigen Siliziumplättchen. Siehe auch 'Chip'.

Intelligentes Datensichtgerät (Intelligent terminal)

Ein Datensichtgerät, bei dem außer der Behandlung von Eingabe und Ausgabe für den Computer auch eine lokale Bearbeitungsmöglichkeit unterstützt wird, auch wenn das Gerät 'offline' ist.

Interaktiv (Interactive)

Dialog zwischen Anwender und Programm, wobei der Anwender aufgefordert wird, etwas einzugeben. Es kann z. B. die Steuerung eines Raumschiffs in einem Automatenspiel als auch das Beantworten von Fragen eines Schulungsprogramms sein. Die Aktion des Anwenders beeinflußt sofort den weiteren Ablauf des Programms.

Interpreter

Eine Erweiterung zur Analogie von Computer Befehlsvorrat und Sprache. Der Interpreter ist eine Systemsoftware, die eine höhere Programmiersprache in die Maschinensprache umwandelt, z. B. konvertiert er BASIC in die interne Sprache des Computers.

Iteration

Eines der Elemente der Datenverarbeitung. Ein Computer führt alle Funktionen durch, indem er sie in einfache kleine Schritte aufteilt, die von der CPU verstanden werden. Der Computer muß dazu zwischen vielen kleinen Schritten hin- und herwechseln, bis eine gewünschte Bedingung erfüllt ist.

K

Die Abkürzung für 'kilo' – 1000. In der Datenverarbeitung aber ist es auch die Abkürzung für Kilobyte – mit einem Wert von 1024, der aus der Sicht des Binärsystems 2 hoch 10 entspricht. Siehe Anhang II.

Kopplung (Handshaking)

Eine Folge von elektronischen Signalen, die den Austausch von Daten zwischen einem Computer und einem peripheren Gerät bzw. zwei Computern anstoßen, prüfen und synchronisieren.

Laderoutine (Booting or Bootstrapping)

Programme und Betriebssysteme werden nicht von selbst geladen, sondern mit einem Ladeprogramm, einer kleinen Routine meist im ROM, die den Ladevorgang auf eine bestimmte Speicheradresse ausstößt.

Leistungstest (Benchmark)

Eine Standardaufgabe für die Ermittlung von Geschwindigkeit, Leistungsfähigkeit und Genauigkeit verschiedener Computer, z. B. die Wurzel aus 99.999 zum Quadrat.

Lichtstift (Light pen)

Eine andere Eingabemöglichkeit, bei der ein Stift oder ein 'Zauberstab' benutzt wird.

Lisp

Die Abkürzung für LISt Processor language, eine weitere höhere Programmiersprache.

Logik (Logic)

Die elektronischen Komponenten, welche die elementaren logischen Operationen und Funktionen ausführen, aus denen jede Operation eines Computers letztendlich aufgebaut ist.

Logo

Eine einfach erlernbare graphikorientierte höhere Programmiersprache, die in Schulen oft im Programmierunterricht verwendet wird.

LSI

Hochgradige Integration (Large Scale Integration). Die Entwicklung von integrierten Schaltungen, bei der immer mehr Funktionen auf immer kleiner werdenden Stückchen Silizium untergebracht werden.

Maschinell lesbar (Machine readable)

Daten in einer Form, die ohne zusätzliche Eingabe über z. B. die Tastatur in den Computer eingegeben werden können.

Maschinencode (Machine Code)

Die Programmiersprache, die der Microprozessor direkt versteht, da alle ihre Kommandos als Bitmuster dargestellt werden.

Matrix (Matrix)

Die Anordnung von Punkten, welche den Bereich eines Schriftzeichens auf dem Bildschirm oder dem Druckkopf eines Punktmatrixdruckers formt. Wird auch in der Mathematik und Informatik zur Beschreibung von Feldern verwendet.

Maus (Mouse)

Ein auf Rollen gleitendes Eingabegerät. Die Maus wird auf dem Tisch hin und her bewegt, um meist den Cursor auf dem Bildschirm zu bewegen. Ursprünglich wurde die Maus dazu entwickelt, um die Angst vor Tastaturen zu nehmen und Software anwenderfreundlicher zu gestalten.

Mensch-Maschinen-Schnittstelle (Man-machine interface)

Der Berührungspunkt zwischen Computer und Anwender: Tastatur, Bildschirm, Geräusche usw.

Menü (Menü)

Eine Auswahlliste der verschiedenen Funktionen, die ein Programm ausführen kann.

Microprozessor (Microprozessor)

Eine integrierte Schaltung im Herzen eines Microcomputers, welche die Maschinencode-Befehle ausführt, die vom BASIC-Interpreter angeboten werden, um die verschiedenen Ausgabegeräte und sonstigen Möglichkeiten zu kontrollieren.

Modem

Ein MOdulator/DEModulator, der den Anschluß eines Computers an eine Telefonleitung oder sonstige serielle Datenübertragungseinrichtung, einschließlich Glasfaserkabel ermöglicht. Siehe auch Akkustikkoppler.

Monitor

Der Bildschirm eines Computerterminals als auch ein Maschinensprachenprogramm, das Zugriffe zu fundamentalen Maschinensprachenoperationen des Computers ermöglicht.

Netz (Network)

Die Verbindung von zwei oder mehr Computern, entweder verdrahtet oder über Modem, um Daten und Informationen untereinander auszutauschen.

Niedere Programmiersprache (Low level language)

Z. B. die Assembler Sprache. Eine Programmiersprache, bei der jede Anweisung mit einer Maschinencode-Anweisung korrespondiert.

Niederwertigstes Bit (Least significant bit)

In einer Binärzahl das am weitesten rechts stehende Bit (siehe Anhang II).

Numerisches Tastenfeld (Numeric keypad)

Ein Teil der Tastatur, in dem Zahlentasten zusammengruppiert sind, um numerische Daten einfacher eintippen zu können. Beim CPC464 können diese Tasten zusätzlich auch als Funktionstasten definiert werden.

OCR

Optical Character Recognition – optische Zeichenerklärung. Ein System bei dem gedruckte oder handgeschriebene Zeichen durch ein optisches Lesegerät erfaßt und direkt in, für den Computer lesbare Daten, umgewandelt werden.

Oktal (Octal)

Das Zahlensystem zur Basis 8, in dem jede Zahl (0-7) mit drei Bits dargestellt wird.

Off line

Ein peripheres Gerät – normalerweise ein Datensichtgerät oder ein Drucker – das nicht betriebsbereit an die Hauptprozessoreinheit angeschlossen ist, auf das also nicht zugegriffen werden kann.

On line

Das Gegenteil von Off line.

Operator

Der Teil eines arithmetischen Ausdrucks, der eine Zahl mit einer anderen verknüpft, wie z. B. + − * usw.

Paddle

Ein anderer Name für Joystick (Steuerknüppel). Auch als 'game paddle' bekannt.

Paperware

Ein anderes Wort für Hardcopy. Manchmal werden Computer, die noch in der Entwicklung sind, 'paperware Übungen' genannt.

Parallele Schnittstelle (Parallel interface)

Die CPC464 Druckerschnittstelle unterstützt einen parallelen Drucker, d. h. jeder Datenausgang des Bus ist mit einem entsprechenden Dateneingang des Druckers verbunden. Daten werden über eine parallele Schnittstelle schneller ausgetauscht als über eine serielle Schnittstelle, da bei serieller Verarbeitung die Daten erst in eine Bytedarstellung mit Synchronisierungsinformation umgeformt werden.

Pascal

Eine strukturierte höhere Programmiersprache, die compiliert werden muß, bevor ein Programm ablaufen kann, aber dann sehr schnell abläuft. Sehr oft die nächste Sprache, die ein eifriger BASIC-Schüler erlernt.

PEEK

Die BASIC-Funktion, die es ermöglicht, den Wert einer bestimmten Stelle des Computerspeichers zu lesen.

Peripheres Gerät (Peripheral)

Drucker, Modems, Joysticks, Plattengeräte – alles was an einen Computer angeschlossen werden kann, um seine Fähigkeiten zu erweitern.

Pixel

Der kleinste Bereich des Bildschirms auf den durch die Hardware bedingt zugegriffen werden kann.

Platte (Disk)

Eine flache, dünne, runde Scheibe aus Kunststoff, die auf einer oder beiden Seiten mit einer magnetisierbaren Beschichtung versehen ist und zur Speicherung von Daten verwendet wird. Die Magnetplatte ist immer in einem Schutzgehäuse untergebracht, das teilweise mit einer Öffnung für den Schreib-/Lesekopf versehen ist. Siehe auch Floppy Disk und Winchester.

Plattengerät (Disk drive)

Die Einheit, die Informationen auf die magnetisierbare Oberfläche der sich drehenden Platte schreibt oder dort vorhandene Informationen wieder liest.

Plotter

Ein Ausgabegerät zum Erstellen von Zeichnungen, das Schreibstifte und nicht einen Druckkopf wie ein Drucker benutzt.

POKE

Die Anweisung in BASIC, womit ein Wert an eine bestimmte Stelle des Speichers geschrieben werden kann. Siehe Kapitel 8.

Port

Eine besonders adressierbare Adresse auf einer Schnittstelle (Interface) für die Eingabe oder Ausgabe von Daten.

Portabilität (Portability)

Die Fähigkeit ein und dieselbe Software auf Computern verschiedener Hersteller ablaufen zu lassen – meist durch kompatible Betriebssysteme erreichbar, wie CP/M von Digital Research.

Programmentwicklung (Software engineering)

Ein Ausdruck für Programmierung mittels Strukturierung und Planung statt mit willkürlichen Techniken.

Puffer (Buffer)

Ein (Zwischen-) Speicher, der vorübergehend Daten aufnimmt, die von einer Funktionseinheit zu einer anderen übertragen werden; z. B. vom Cassettenrecorder zur Zentralprozessoreinheit (CPU) und zum Haupt-RAM. Der Puffer reguliert den Datenfluß zwischen Einheiten, die mit verschiedenen Geschwindigkeiten arbeiten, wie z. B. einem Modem oder einem Drucker.

Punktmatrix (Dot matrix)

Ein rechteckiges Punktraster, auf dem ein Zeichen durch Auswahl einzelner Punkte dargestellt werden kann.

RAM

Random Access Memory. Eine Speichereinheit, die mit der internen Schaltung des Computers während einer normalen Programmausführung sowohl gelesen als auch beschrieben werden kann.

Raster

Eine Art des 'Schreibens' auf dem Bildschirm, wobei die Bilder aus einer Anzahl waagerechter Abtastlinien aufgebaut werden.

Reelle Zahl (Real number)

Eine Zahl mit ganzzahligem und Bruchteil, also mit Werten zu beiden Seiten des Dezimalpunktes. Eine Variable kann trotzdem eine reelle Zahl sein, obwohl sie manchmal im Programm nur den ganzzahligen Teil besitzt.

Register

Eine Speichereinheit in der CPU, die dazu benutzt wird, Daten kurzzeitig zwischenzuspeichern.

Rekursion

Eine Serie von wiederholten Schritten innerhalb eines Programms oder einer Routine, deren Ergebnisse bei jeder Wiederholung in Verbindung zu denen des vorangegangenen Durchlaufs stehen.

Reserviertes Wort (Reserved word)

Ein Wort mit besonderer Bedeutung in der Programmiersprache, das nicht anders als vordefiniert verwendet werden kann. Z. B. wird BASIC das Wort NEW nicht als Variablenname annehmen, da es schon für einen anderen Zweck reserviert ist.

RF Modulator

Die Art, in der Videosignale des Computers umgeschlüsselt werden, so daß sie über den Antenneneingang eines Fernsehers 'gesendet' werden können.

ROM

Nur lesbare Speicher (Read Only Memory). Halbleiterspeicher, der einmal beschrieben, weder gelöscht noch überschrieben werden kann.

RS232C

Ein spezieller Standard für serielle Datenübertragungsschnittstellen. Geräte an beiden Enden der Übertragungsstrecke müssen an die spezifischen Bedingungen der benutzten RS232-Schnittstelle angepaßt sein. Auch die Centronics-(parallel)Schnittstelle ist ja engen Regeln unterworfen.

Schleife (Loop)

Ein Prozess im Programm, der sooft wiederholt wird, bis eine bestimmte Bedingung erfüllt ist.

Schlüsselwort (Keyword)

Ein Wort, dessen Anwendung in einem Computerprogramm oder einer Sprache für eine spezifische Funktion oder eines Kommandos reserviert ist.

Schnittstelle (Interface)

Der Ein- und Ausgang eines Computers sowohl in elektrischer als auch in menschlicher Hinsicht. Als Schnittstelle am CPC464 gelten die Tastatur (Eingabe), der Bildschirm (Ausgabe) genauso wie der Schnittstellenbus auf der Computerrückseite für periphere Geräte.

Schreibmaschinentastatur (QWERTY keyboard)

Umgangssprachlicher Begriff für eine Computertastatur, die in ihrer Tastenanordnung wie eine Schreibmaschine aufgebaut ist.

Scrolling

Die Art, wie die Zeilen des Bildschirms nach oben 'gerollt' werden, um Platz in der untersten Zeile für neue Ein- oder Ausgaben zu machen.

Serielle Schnittstelle (Serial interface)

Dieser Begriff bezieht sich fast immer auf eine RS232-Schnittstelle, obwohl noch andere serielle Schnittstellen für sequentielle Datenübertragung existieren.

Simulation

Eine Technik, lebensnahe Aktionen mit Hilfe des Computers nachzuahmen, wie z. B. Flug- und Fahrsimulation usw.

Soft Key

Siehe Benutzerdefinierte Taste (User defined key-UDK).

Software

Programme.

Speicher (Memory)

Der Bereich eines Computers, in dem Informationen und Daten geordnet zwischengelagert werden, so daß auf alles wieder zugegriffen werden kann; bekannt entweder als RAM (random access memory – wahlfreier Zugriffsspeicher), der sowohl beschrieben als auch gelesen werden kann, oder als ROM (read only memory – nur lesbarer Speicher), der nur gelesen aber nicht beschrieben werden kann. Magnetplatten und -bänder sind Beispiele für 'Massenspeicher', wobei dieser Begriff heute öfter für den Speicherbereich verwendet wird, der direkt von der CPU angesprochen wird.

Speicherbelegungsplan (Memory map)

Die Beschreibung der Speichernutzung mit den jeweiligen Adressen und der Speicherzuordnung für die verschiedenen Funktionen, wie z. B. für den Bildschirm, das Magnetbandbehandlungssystem usw.

Spracherkennung (Speech reconition)

Die Umwandlung gesprochener Wörter in Anweisungen, die der Rechner erkennen kann.

Sprachsynthese (Speech synthesis)

Die Erzeugung von simulierter Sprache mittels Hard- und Software.

Sprite

Ein Zeichen, von spezieller Hard- oder Software erzeugt, das sich frei auf dem Bildschirm bewegt und zufällig erscheinen oder verschwinden kann.

Stapel (Stack)

Ein Bereich im Speicher zum Ablegen von Informationen, woraus aber nur der jeweils letztgeschriebene Eintrag wieder geholt werden kann.

Steckmodul (Cartridge)

Eine besonders verpackte Speichereinheit mit integrierten Schaltungen, die Software enthält und durch dafür eingebaute Stecksockel direkt an den Computer angeschlossen werden kann. Diese Software wird schneller geladen und läuft auch schneller ab als Software von Cassetten, ist jedoch wesentlich teuerer.

Steuerknüppel (Joystick)

Ein Eingabegerät, das die Funktionen der Cursorsteuertasten ersetzt, um Spiele schneller zu machen.

Strichcode (Bar code)

Beispiele finden Sie auf vielen Lebensmittelverpackungen. Ein Code aus Linien unterschiedlicher Stärke, die mit einer Lesevorrichtung (z. B. Schwachstromlaser) erfaßt und ausgewertet werden.

Strukturierte Programmierung (Structured programming)

Eine logische und vorbedachte Art der Programmierung, deren Ergebnis Programme sind, die mit klar definierten Schritten von 'oben nach unten' ablaufen.

Syntaxfehler (Syntax error)

Wenn der Ablauf eines Programms durch eine falsche Benutzung von Schlüsselwörtern und Variablen abgebrochen wird, gibt BASIC diese Meldung für den Anwender aus. Siehe Anhang VIII.

Tabellenkalkulation (Spreadsheet)

Ein Programm, das in Zeilen und Spalten eingetragene Zahlen arithmetisch behandeln kann. Wenn ein Eintrag geändert wird, werden alle damit verbundenen Kalkulationen neu errechnet und ein neues Ergebnis errechnet.

Tastatur (Keyboard)

Eine Zusammenstellung von alphanumerischen Tastenschaltern, um Kommandos und andere Informationen in den Computer eingeben zu können.

Teilprogramm (Routine)

Ein Teil des Programms, in dem ein zusammengehörender Teil der Programmlogik bearbeitet wird. Ein solches Teilprogramm kann im Hauptprogramm untergebracht oder auch ein eigenes Unterprogramm sein, das in verschiedenen anderen Programmen mitverwendet werden kann. Z. B. kann man ein Programm, in dem die Uhrzeit aus der Systemuhr in eine normale 12-Stundenuhr aufbereitet wird, als Teilprogramm gestalten.

Terminal

Ein peripheres Gerät, bestehend aus einer Tastatur und entweder einem Bildschirm oder einem Fernschreiber als Ausgabesystem.

Textfolge (String)

Die Arten von Daten, die aus einer Folge von Zeichen bestehen und nicht als numerische Werte betrachtet werden. Auch wenn Sie aus nur numerischen Zeichen bestehen, so werden sie nicht als numerisch betrachtet, solange sie nicht durch ein entsprechendes Umwandlungskommando einer numerischen Variablen zugewiesen worden sind.

Tongenerator (Sound generator)

Der Teil des Computer (es kann sowohl Hard- als auch Software sein), der Töne und Geräusche erzeugt.

Trennzeichen (Separator)

Ein Trennzeichen hat die gleiche Wirkung wie ein Begrenzer (Delimiter): Die Grenze zwischen reservierten Wörtern und sonstigen Elementen des Programms oder der Daten wird hiermit markiert.

Typenrad-Drucker (Daisy-wheel printer)

Ein Drucker mit hoher Schreibqualität. Zeichen werden durch Anschlag eines Buchstabens auf einem Rädchen gegen ein Farbband erzeugt.

Überschreiben (Overwrite)

Löschen eines Bereichs im Speicher durch Überschreiben seines Inhalts mit neuen Daten.

Uhr (Clock)

Ein Zeitsystem im Computer als Basis für den Gleichlauf der internen Funktionen. Eine Echtzeituhr ist eine, die Stunde, Datum usw. verwaltet.

Umgekehrte polnische Notation (Reverse Polish notation – RPN)

Eine Methode, die manche Hersteller bei der Errechnung von arithmetischen Ausdrükken in ihren Computern verwenden, wobei die Operatoren hinter den Werten angeordnet werden, zu denen sie gehören.

Unintelligentes Datensichtgerät (Dumb terminal)

Ein Datensichtgerät, das nur als Mittel zur Dateneingabe bzw. -ausgabe dient, ohne die Daten irgendwie zu bearbeiten. Es gibt noch einfachere Sichtgeräte, die nicht einmal einen Generator haben, so daß alle Abbildungen am Bildschirm reine Videodarstellungen sind.

Unterprogramm (Subroutine)

Siehe Teilprogramm (Routine)

Variable

Ein Datenfeld in einem Programm, das mit einem Namen angesprochen werden und dessen Wert während des Programmablaufs verändert werden kann.

Vorzeichenlose Zahl (Unsigned number)

Eine Zahl ohne ein vorangestelltes Kennzeichen, das aussagt ob die Zahl positiv oder negativ ist.

Wahrheitstabelle (Truth table)

Das Ergebnis einer logischen Operation ist entweder 'wahr' oder 'falsch'. Der Computer stellt dies entweder mit 1 oder 0 dar. Eine Liste aller möglichen Ergebnisse einer logischen Operation (z. B. IF A > B THEN C) heißt Wahrheitstabelle.

Wiederauffrischen (Refresh)

Das Aktualisieren von Daten auf einen Bildschirm oder im Speicher. Die schon vorhandenen Daten müssen dabei nicht unbedingt zerstört werden, sondern können auch nur wieder aufgefrischt werden.

Zeichen (Character)

Ein Buchstabe, eine Nummer oder ein graphisches Symbol, das im Computer dargestellt und ausgegeben werden kann (siehe Anhang III).

Zeichenkette (Character String)

Eine Folge von Zeichen, die als Einheit gespeichert oder behandelt werden können, z. B. ein Wort oder eine Kette von Worten.

Zeichenvorrat (Charcter set)

Alle Buchstaben, Nummern und graphische Symbole, die in einem Computer oder auf einem Drucker zur Verfügung stehen. Daß ein Zeichen im Computer zur Verfügung steht, bedeutet nicht, daß es auch auf einem Drucker vorhanden ist.

Zeichenzelle (Charakter cell)

Die Punktematrix auf dem Bildschirm, in der ein Zeichen durch unterschiedliche Farbgebung der einzelnen Punkte sichtbar gemacht wird.

Zeilennummer (Line number)

BASIC und einige andere Sprachen benutzen die Zeilennummer als Ordnungskriterium innerhalb des Programms.

Zufallszahl (Random number)

Eine vom Computer generierte Zahl, die weder wiederholbar noch voraussagbar ist.

Anhang II

Eine Einführung in die Hintergründe der Datenverarbeitung

Wer hat Angst vor der Fachsprache?

Wie in allen 'spezialisierten' Industrien hat die Datenverarbeitung eine eigene Fachsprache entwickelt, eine Art Kurzschrift für die Übermittlung komplizierter Begriffe, die sonst viele 'Klartext'-Worte in Anspruch nehmen würden. Es sind nicht nur die hochtechnisierten Arbeitsgebiete, die sich hinter einer Mauer von 'Schlagwörtern', Fachsprache und Spezialbegriffen verstecken – die meisten Berufe und Arbeitsgebiete haben ihre eigene 'Sprache', wogegen wir alle schon einmal zu kämpfen hatten.

Der größte Unterschied ist, wie bei den Juristen, daß die benutzten Wörter nicht so schwierig sind, nur die Art, wie sie gebraucht werden. Die Leute, die mit der Datenverarbeitung groß geworden sind, versuchen diese Begriffe unkompliziert anzuwenden, um so die Komplexität zu verkleinern.

Lassen Sie sich nicht durch die 'einfache Sprache' in der Datenverarbeitung in die Irre führen, sie ist kein literarisches Thema, sondern eine präzise Wissenschaft. Abgesehen von der 'Syntax' der Sprache, ist ihre Struktur sehr einfach und absolut nicht verwirrend oder widersprüchlich. Lehrer der Datenverarbeitung haben es bisher noch nicht versucht, eine Art Standard zur genauen Analysierung eines Programms zu entwickeln.

Trotzdem können Programme analysiert werden. Obwohl die Bedeutung eines Programmes nicht auf Anhieb klar sein kann, sticht sofort ins Auge, ob das Programm elegant oder unübersichtlich ist. Heute wird wesentlich mehr Nachdruck auf ordentlichen Programmaufbau gelegt, als es während des Anfangsdurcheinanders der Micro-Revolution üblich war.

Die Datenverarbeitung wird schnell von vielen jungen Leuten verstanden, da sie für die Genauigkeit und Einfachheit der Ideen einen Sinn haben, ebenso wie für die Art, wie sie weitergegeben werden. Sie werden kaum einen zehnjährigen Anwalt finden – aber Sie finden sehr viele zehnjährige Programmierer!

Bild Appendix II Page 1

Grundlagen des BASIC

Fast alle Heimcomputer bevorzugen als Programmiersprache BASIC, womit Programme in fast normaler Sprache geschrieben werden können. BASIC bedeutet 'Beginners All-purpose Symbolic Instruction Code' (allgemeiner symbolischer Befehlscode für Anfänger), was nicht heißen soll, daß BASIC primitiv ist – viele sehr komplexe und leistungsfähige Programme werden in BASIC geschrieben.

Es kann aber nicht daran gezweifelt werden, daß der Name BASIC viele Anfänger für sich gewonnen hat, da er in dem Wirrwarr der vielen vorhandenen Programmiersprachen vielversprechender klingt. Dies hat wahrscheinlich zu der weiten Verbreitung von BASIC geführt.

Ab jetzt werden die Begriffe der Computersprache bei ihrem ersten Auftreten durch Kursivschrift kenntlich gemacht. Haben Sie keine Angst vor dem Versuch, die Begriffe in diesem Abschnitt zu lernen – es gibt noch ein Verzeichnis aller Begriffserklärungen in einem eigenen Kapitel.

Grundlagen

BASIC ist eine Programmiersprache, die den Umfang der erlaubten Kommandos interpretiert und dann Operationen an Daten während des Programmablaufs ausführt. Während der menschliche Wortschatz ca. 5- bis 8-tausend Worte umfaßt (zuzüglich der verschiedenen Anwendungen von Verben usw.), kommt BASIC mit ca. 200 Worten aus. Programme, die mit BASIC geschrieben werden, unterliegen den starren Regeln für die Anwendung dieser Worte. Die *Syntax* ist sehr präzise, und jeder Versuch, eine andere Schreib- oder Sprachweise zu verwenden, endet mit der kalten und nüchternen Meldung:

```
Syntax error
```

Dies ist nicht so einschränkend, wie es auf den ersten Blick erscheint, da die BASIC Sprache (Syntax) hauptsächlich für die Bearbeitung numerischer Daten gedacht ist. Die Worte sind im wesentlichen nur Erweiterungen der bekannten mathematischen Operationen wie +/− usw. Das Wichtigste für einen Anfänger ist es, zu akzeptieren, daß der Computer nur mit numerischen Daten arbeiten kann, da jede Information nur in der Form von numerischen Daten an die *integrierte Schaltung des zentralen Prozessors* übergeben werden kann.

Zahlen bitte!

Würde ein Computer dazu benutzt, die kompletten Werke von Goethe zu speichern, dann wäre kein einziger Buchstabe oder kein einziges Wort im ganzen System zu finden. Jedes Stück der Information wird erst in eine Zahl umgewandelt, mit welcher der Computer arbeiten kann.

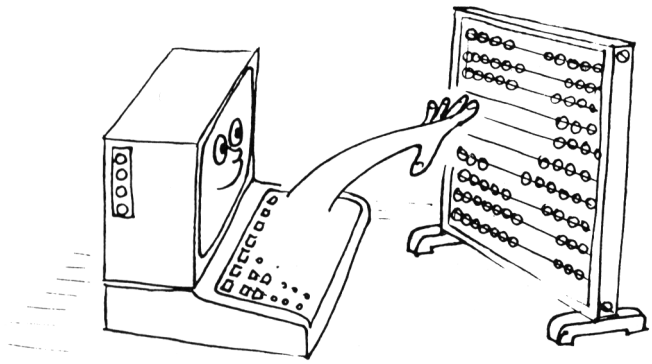
BASIC interpretiert die Worte als Zahlen, die der Computer dann mit Addition, Subtraktion und Eigenschaften der *Booleschen Logik* behandeln kann, so daß er die Daten vergleichen und einige Teile davon auswählen kann, um z. B. zu prüfen, ob eine Zahl größer als die andere ist oder um einen bestimmten Ablauf einzuleiten, falls die eine ODER andere Zahl dies verlangt.

Mittels des Programms wird jede Aufgabe vom Computer in eine Serie von einfachen Ja/Nein Operationen zerlegt.

Die Multiplikation wird in der Form mehrerer Additionen durchgeführt. Die BASIC-Anweisung zur Multiplikation von 35 mit 10 (35*10) wird ausgeführt, indem 35 zehnmal auf sich selbst addiert wird.

In einen Bereich des *Zentralen Prozessors (CPU)* wird die Zahl 10 geladen, in einen an deren die Zahl 35. Immer wenn 35 auf sich selbst addiert wird, wird der Bereich im <Speicher> mit dem Inhalt 10 um 1 vermindert bis er 0 wird, die 'Multiplikation' damit endet und das summierte Ergebnis 350 in einen anderen Bereich der CPU, zur *Ausgabe*, als Antwort übertragen wird.

Bild Appendix II Page 3



Wenn Ihnen dieser Prozeß umständlich vorkommt, dann haben Sie recht und Sie haben die erste und wichtigste Wahrheit der Datenverarbeitung entdeckt. Ein Computer ist ein Werkzeug zur Ausführung allereinfachster, immer wiederkehrender Aufgaben – sehr schnell und absolut präzise. Also, BASIC interpretiert Anweisungen in einem Programm und wandelt Sie in eine Sprache um, welche die *CPU* behandeln kann. Der Computer versteht nur zwei Zustände: 'Ja' und 'Nein', die in *binärer* Schreibweise mit '1' oder '0' dargestellt werden. Es gibt in der *Booleschen Logik* nur 'wahr' oder 'falsch', kein 'vielleicht'.

Das Umschalten zwischen diesen beiden Zuständen ist die wesentlichste Bedeutung von *digital*. In der Natur bewegen sich Prozesse nur allmählich von einem 'stabilen' Zustand zum anderen in einer glatten Weise. Mit anderen Worten, der Übergang erfolgt hierbei auf einem langen Weg zwischen den beiden Zuständen, in einer reinen digitalen Umgebung nimmt dieser Übergang keine Zeit in Anspruch – wobei jedoch physikalische Grenzen der Microelektronik eine minimale Verzögerung verursachen, die sogenannte Laufzeitverzögerung. Die Summe vieler dieser Laufzeitverzögerungen ist der Grund, warum es etwas Zeit dauert, bis die Information bearbeitet ist und eine Antwort ausgegeben wird.

In jedem Fall muß der Computer eine bestimmte Zeit warten, bis eine Aufgabe fertig ist und er deren Ergebnis für weitere Aufgaben weiterverwenden kann. Ein digitaler Prozeß kennt nur schwarz oder weiß, wobei der Übergang in Grautönen bedeutungslos ist. Der allmähliche Übergang führt durch alle Grautöne.

Wenn die letztendliche Antwort nur 0 oder 1 sein kann, so kann es keine 'fast' richtige Antwort geben. Daß es manchmal so aussieht, als ob der Computer Fehler bei der Behandlung von numerischen Daten macht, hängt damit zusammen, daß der Computer in der Größe der Darstellung einer Zahl begrenzt ist. Numerische daten, die diese Grenzen überschreiten, werden *abgeschnitten*, was zu Rundungsfehlern führen kann. Dadurch kann 999,999,999 dann 1,000,000,000 werden.

Bits und Bytes

Wir haben uns daran gewöhnt, mit Zahlen im *Dezimal*-System umzugehen, wobei der Bezugspunkt die Zahl 10 ist, d. h. Werte werden mit zehn Ziffern im Bereich von 0 bis 9 dargestellt (0 bis 9 ist der Schreibweise von 1 bis 10 vorzuziehen). Das Zahlensystem mit Zahlen im Bereich von 0 bis 1 ist das *Binär*-System, und die Einheiten, womit das System arbeitet, heißen Bits, kurz für 'BInary DigiT'. Es könnte weniger verwirrend sein, wenn Sie 0 und 1 im Zusammenhang mit dem Dezimal-System vergessen und zwei andere Symbole für die gleichen Funktionen nehmen: fl und * könnten genauso verwendet werden.

Der Zusammenhang zwischen *Bits* und der dezimalen Schreibweise ist einfach zu verstehen:

Es ist normal, die maximale Anzahl von Binärziffern im Gebrauch anzugeben und führende Nullen zu schreiben, so daß immer die ganze Anzahl der Bits verwendet wird:

dezimal 7 wird
00111 Binär in 5-Bit-Schreibweise.

Im Binärsystem können die Ziffern als Anzeigen in Spalten verstanden werden, die aussagen, ob die entsprechenden Potenzen von 2 vorhanden (1) sind oder nicht (0).

$2^0 = 1$
$2^1 = 2 = 2 = 2(2^0)$
$2^2 = 4 = 2\times2 = 2(2^1)$
$2^3 = 8 = 2\times2\times2 = 2(2^2)$
$2^4 = 16 = 2\times2\times2\times2 = 2(2^3)$

so daß die Spalten wie folgt aussehen können

2^4	2^3	2^2	2^1	2^0
1	0	0	1	1
(16	0	0	2	1) = 19

Eine Kurzschreibweise, um Binärziffern darstellen zu können ist der Begriff 'Byte' für 8 Bits. Die größte Zahl, die in einem Byte gespeichert werden kann ist (Binär) 11111111 – oder (Dezimal) 255. Das ergibt 256 echte Variationen, einschließlich 00000000 (auch ein zulässiger Datenwert für den Computer).

Computer behandeln Daten in Vielfachen von 8 Bits. 256 ist keine so große Zahl, so daß zur Adressierung des Speichers zwei Bytes verwendet werden. Der Speicher kann als *Bereich* mit einer horizontalen und vertikalen *Adresse* betrachtet werden, in dem jedes Element folgendermaßen angesprochen werden kann:

0	1	2	3	4	5	6	7	8	9
1									
2									
3									
4									
5				1		1			
6									
7									
8									
9									

Dieser Bereich kann bis zu (10 × 10) Informationseinheiten aufnehmen, deren Adressen jeweils im Bereich von 0 bis 9 liegen. Auf der Position 3,5 ist '1' gespeichert, auch auf 5,5.

Ein Bereich mit 256 × 256 Einheiten umfaßt 65,536 einzelne Positionen, die mit 8-Bit-Adressen für die waagrechten und die senkrechten Achsen angesprochen werden können. Mit unseren '0' und '1' können wir nun 65,536 Elemente adressieren.

Die nächste Ebene der Kurzschreibweise ist Kilobyte (k-Byte oder einfach '*K*'), was 1024 Bytes bedeutet. 1024 ist das dem dezimalen Begriff 'kilo' nächstgelegene binäre Vielfache und erklärt, warum ein Computer mit '64K' Speicher in Wirklichkeit mit 65.536 Bytes (64 × 1024) Speicher ausgestattet ist.

Glücklicherweise führt der BASIC Interpreter alle notwendigen Konvertierungen für Sie durch, so daß Sie ein fähiger Programmierer werden können, ohne das Binärsystem vollständig verstehen zu müssen. Sollten Sie das Binärsystem voll begriffen haben, wird es Ihnen helfen, schneller die 'mysteriösen' bzw. signifikanten Zahlen zu verstehen, die unvermeidlich auftauchen werden, wenn Sie sich durch die Wissenschaft der Datenverarbeitung durcharbeiten.

Es lohnt sich, die Mühe aufzuwenden, um das Binärsystem zu verstehen und die verschiedenen Zahlen wie 255, 1024, usw. zu erkennen, da es sehr unwahrscheinlich ist, daß sich diese Grundlagen der Computerarbeitsweise in absehbarer Zeit ändern werden. Die Sicherheit und Einfachheit, die durch die Arbeit mit nur zwei Zuständen entsteht, wird über die sich steigernde Komplexität siegen, die mit jedem anderen Zahlensystem gegeben wäre.

Jedoch...

Die binäre Schreibweise ist denkbar einfach und elegant, aber sie ist aufwendig zu schreiben und verleitet zu Fehlern beim Lesen. Das Binärsystem ist mit anderen Zahlensystemen verwandt, die als Kurzschreibweise für Programmierer dienen. Ein solches Zahlensystem, das im Zusammenhang mit Microcomputern verwendet wird, heißt *Hex* (Kurzform von Hexadezimal).

Dabei basieren die Zahlen auf der Zahl 16 und werden mit einem einzigen Zeichen dargestellt.

Dezimal

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Hex

0 1 2 3 4 5 6 7 8 9 A B C D E F

Das Hexadezimalsystem kann die acht Bits eines Bytes in zwei Blöcke zu je vier Bits aufteilen, da 15 eine *vier Bit* Zahl ist: 1111 binär. Der erste Block enthält die Anzahl kompletter 15-er Einheiten, der zweite Block enthält den 'Divisionsrest' – hierbei kommt die ganze Eleganz des Binärsystems zum Tragen.

Hier nochmal die Tabelle zur Einführung in die Binärschreibweise:

Dezimal	Binär	Hexadezimal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Eine 8-Bit Zahl 11010110 kann in zwei 4-Bit Zahlen, genannt *Halbbytes*, unterteilt werden, Hex D6. In diesem Handbuch werden Hex-Zahlen immer mit dem Symbol '&' gekennzeichnet, z. B. &D6, und dieses Zahlensystem wird auch von den meisten Programmierern in einer Assembler-Sprache benutzt. Die Assembler-Sprache bietet den engsten Kontakt zum *Maschinencode* überhaupt, da hierbei einfache Buchstabenkurzzeichen anstelle der echten Maschinencodezahlen benutzt werden.

Wenn Sie mit HEX arbeiten, müssen Sie die erste Ziffer zur Bestimmung der Anzahl der 16-er im endgültigen Ergebnis verwenden, dann die zweite Ziffer dazu addieren, um den endgültigen Dezimalwert auszurechnen. Es gibt eine starke Neigung, eine Zahl wie &D6 als 13+6 oder 136 zu betrachten. Aber richtig ist $(13 \times 16)+(16) = 214$.

Es ist genauso wie beim Dezimalsystem, in dem man z. B. für 89 liest: $(8 \times 10) + (9)$. Die Multiplikation mit 10 kommt Ihnen einfacher vor, weil Sie viel mehr Übung damit haben als mit 16.

Wenn Sie soweit gekommen sind, ohne sehr durcheinander zu sein, dann haben Sie einen guten Start gemacht, um die Grundlagen des Computers zu begreifen. Und wenn Sie sich fragen, wozu das alles – dann haben Sie recht. Ein Computer ist eine Maschine, die mit sehr einfachen Ideen und Konzepten arbeitet, aber das sehr schnell (Millionen von Malen je Sekunde) und mit einer enormen Kapazität, sich beides zu merken, welche Daten eingegeben wurden, sowie die Abertausend von Zwischensummen, die auf dem Weg zum endgültigen Ergebnis entstehen.

Um die Theorie über Ihren Computer weiter zu vertiefen, stehen Tausende von Büchern zur Verfügung. Einige werden Sie nur verwirren, andere dagegen werden Sie sehr viel weiter bringen und Ihnen enthüllen, wie schlicht und einfach die grundlegenden Zusammenhänge zwischen Zahlensystemen und deren Behandlung in Ihrem Computer sind.

Bild App. II/S.7

Anhang III

Die ASCII Zeichen und die graphischen Zeichen des CPC464

III. 1 ASCII

Als praktische Hilfe hier eine Zusammenstellung des Standard ASCII Zeichensatzes mit dezimaler, oktaler und hexadezimaler Schreibweise zusammen mit dem entsprechenden ASCII-Code. Jedes der CPC464 Zeichen ist auch im Detail abgebildet.

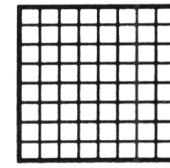
DEC	OCTAL	HEX	ASCII characters
0	000	00	NUL ([CTRL]A)
1	001	01	SOH ([CTRL]B)
2	002	02	STX ([CTRL]C)
3	003	03	ETX ([CTRL]C)
4	004	04	EOT ([CTRL]D)
5	005	05	ENQ ([CTRL]E)
6	006	06	ACK ([CTRL]F)
7	007	07	BEL ([CTRL]G)
8	010	08	BS ([CTRL]H)
9	011	09	HT ([CTRL]I)
10	012	OA	LF ([CTRL]J)
11	013	OB	VT ([CTRL]K)
12	014	OC	FF ([CTRL]L)
13	015	OD	CR ([CTRL]M)
14	016	OE	SO ([CTRL]N)
15	017	OF	SI ([CTRL]O)
16	020	10	DLE ([CTRL]P)
17	021	11	DC1 ([CTRL]Q)
18	022	12	DC2 ([CTRL]R)
19	023	13	DC3 ([CTRL]S)
20	024	14	DC4 ([CTRL]T)
21	025	15	NAK ([CTRL]U)
22	026	16	SYN ([CTRL]V)
23	027	17	ETB ([CTRL]W)
24	030	18	CAN ([CTRL]X)
25	031	19	EM ([CTRL]Y)
26	032	1A	SUB ([CTRL]Z)
27	033	1B	ESC
28	034	1C	FS
29	035	1D	GS
30	036	1E	RS
31	037	1F	US
32	040	20	SP
33	041	21	!
34	042	22	"
35	043	23	#
36	044	24	$
37	045	25	%
38	046	26	&
39	047	27	'
40	050	28	(
41	051	29	)
42	052	2A	*
43	053	2B	+
44	054	2C	,
45	055	2D	-
46	056	2E	.
47	057	2F	/
48	060	30	Ø
49	061	31	1

DEC	OCTAL	HEX	ASCII
50	062	32	2
51	063	33	3
52	064	34	4
53	065	35	5
54	066	36	6
55	067	37	7
56	070	38	8
57	071	39	9
58	072	3A	:
59	073	3B	;
60	074	3C	<
61	075	3D	=
62	076	3E	>
63	077	3F	?
64	100	40	@
65	101	41	A
66	102	42	B
67	103	43	C
68	104	44	D
69	105	45	E
70	106	46	F
71	107	47	G
72	110	48	H
73	111	49	I
74	112	4A	J
75	113	4B	K
76	114	4C	L
77	115	4D	M
78	116	4E	N
79	117	4F	O
80	120	50	P
81	121	51	Q
82	122	52	R
83	123	53	S
84	124	54	T
85	125	55	U
86	126	56	V
87	127	57	W
88	130	58	X
89	131	59	Y
90	132	5A	Z
91	133	5B	[
92	134	5C	\
93	135	5D	]
94	136	5E	↑
95	137	5F	_
96	140	60	
97	141	61	a
98	142	62	b
99	143	63	c

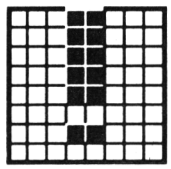
DEC	OCTAL	HEX	ASCII
100	144	64	d
101	145	65	e
102	146	66	f
103	147	67	g
104	150	68	h
105	151	69	i
106	152	6A	j
107	153	6B	k
108	154	6C	l
109	155	6D	m
110	156	6E	n
111	157	6F	o
112	160	70	p
113	161	71	q
114	162	72	r
115	163	73	s
116	164	74	t
117	165	75	u
118	166	76	v
119	167	77	w
120	170	78	x
121	171	79	y
122	172	7A	z
123	173	7B	{
124	174	7C	|
125	175	7D	}
126	176	7E	~

III.2 CPC464 Maschinenspezifischer Firmware Zeichensatz

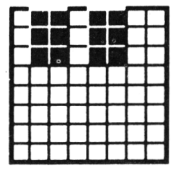
Die hier abgebildeten Zeichen sind in einer 8×8 Matrix dargestellt, wie sie auf dem CPC464 Bildschirm erscheinen. Anwendereigene Zeichen können für spezielle Effekte zusammengestellt werden und bündig aneinanderstoßen – siehe `SYMBOL`-Kommando in Kapitel 8.



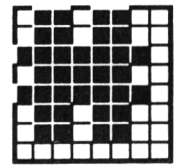
32 &H20
&X00100000



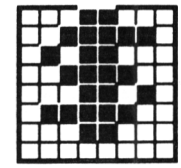
33
&H21
&X00100001



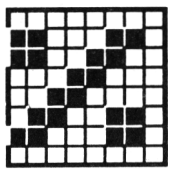
34
&H22
&X00100010



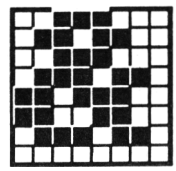
35
&H23
&X00100011



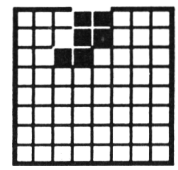
36
&H24
&X00100100



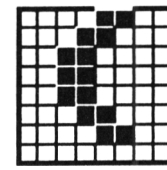
37
&H25
&X00100101



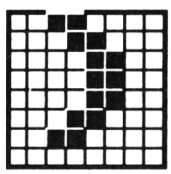
38
&H26
&X00100110



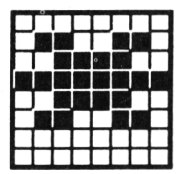
39
&H27
&X00100111



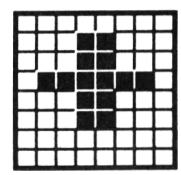
40
&H28
&X00101000



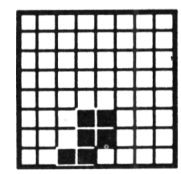
41
&H29
&X00101001



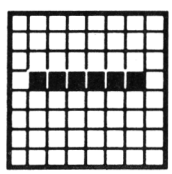
42
&H2A
&X00101010



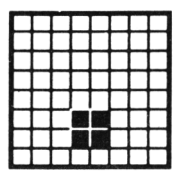
43
&H2B
&X00101011



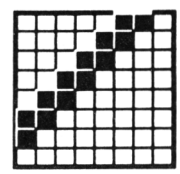
44
&H2C
&X00101100



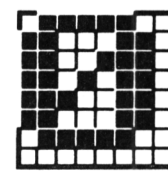
45
&H2D
&X00101101



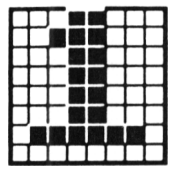
46
&H2E
&X00101110



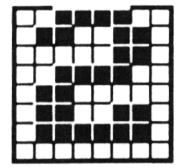
47
&H2F
&X00101111



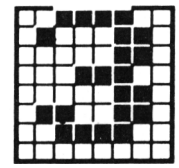
48
&H30
&X00110000



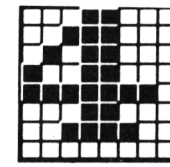
49
&H31
&X00110001



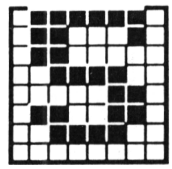
50
&H32
&X00110010



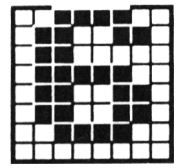
51
&H33
&X00110011



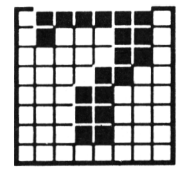
52
&H34
&X00110100



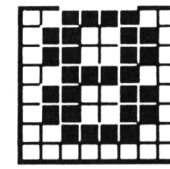
53
&H35
&X00110101



54
&H36
&X00110110

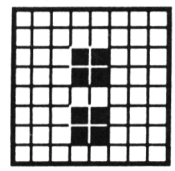


55
&H37
&X00110111

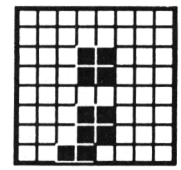


56
&H38
&X00111000

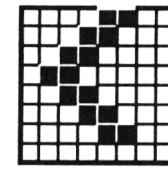
57
&H39
&X00111001



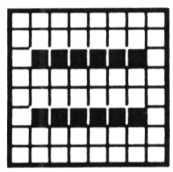
58
&H3A
&X00111010



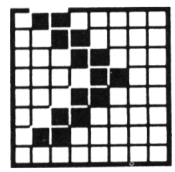
59
&H3B
&X00111011



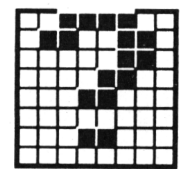
60
&H3C
&X00111100



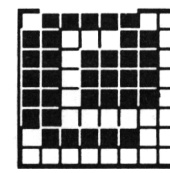
61
&H3D
&X00111101



62
&H3E
&X00111110

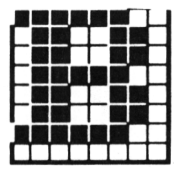


63
&H3F
&X00111111

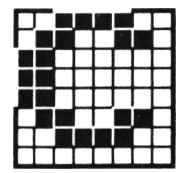


64
&H40
&X01000000

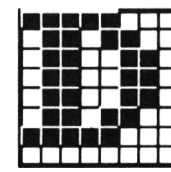
65
&H41
&X01000001



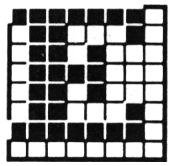
66
&H42
&X01000010



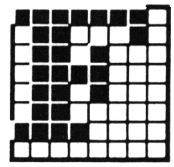
67
&H43
&X01000011



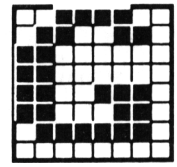
68
&H44
&X01000100



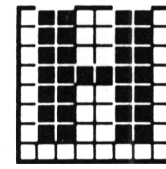
69
&H45
&X01000101



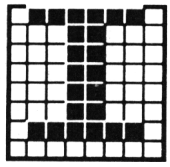
70
&H46
&X01000110



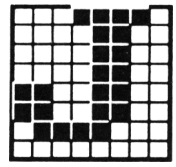
71
&H47
&X01000111



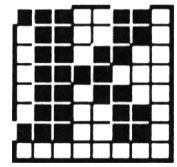
72
&H48
&X01001000



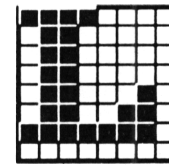
73
&H49
&X01001001



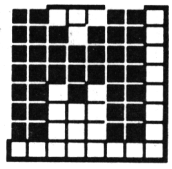
74
&H4A
&X01001010



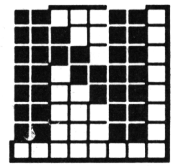
75
&H4B
&X01001011



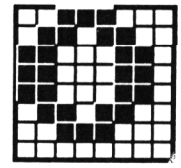
76
&H4C
&X01001100



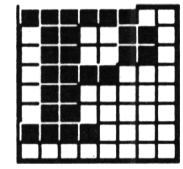
77
&H4D
&X01001101



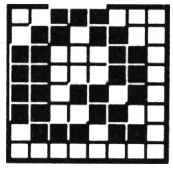
78
&H4E
&X01001110



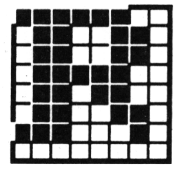
79
&H4F
&X01001111



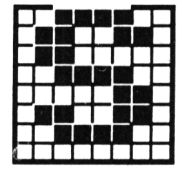
80
&H50
&X01010000



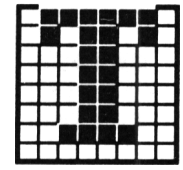
81
&H51
&X01010001



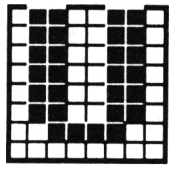
82
&H52
&X01010010



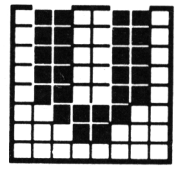
83
&H53
&X01010011



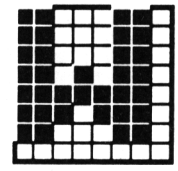
84
&H54
&X01010100



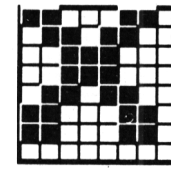
85
&H55
&X01010101



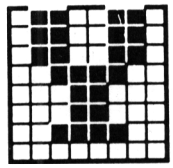
86
&H56
&X01010110



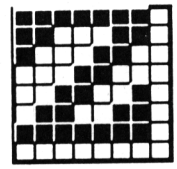
87
&H57
&X01010111



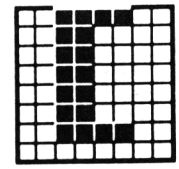
88
&H58
&X01011000



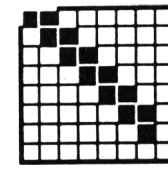
89
&H59
&X01011001



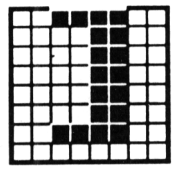
90
&H5A
&X01011010



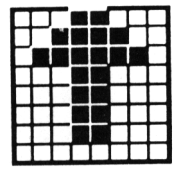
91
&H5B
&X01011011



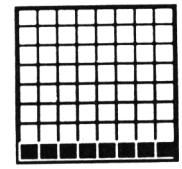
92
&H5C
&X01011100



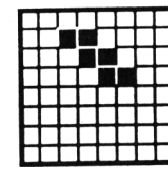
93
&H5D
&X01011101



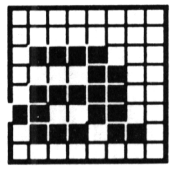
94
&H5E
&X01011110



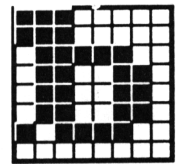
95
&H5F
&X01011111



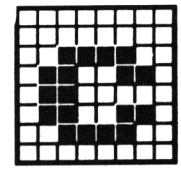
96
&H60
&X01100000



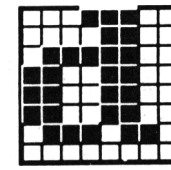
97
&H61
&X01100001



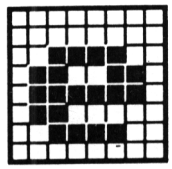
98
&H62
&X01100010



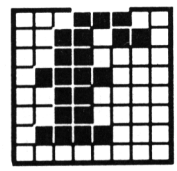
99
&H63
&X01100011



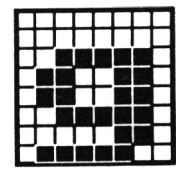
100
&H64
&X01100100



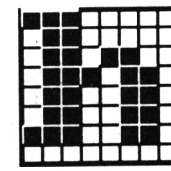
101
&H65
&X01100101



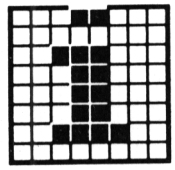
102
&H66
&X01100110



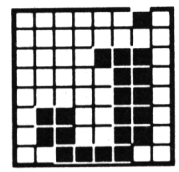
103
&H67
&X01100111



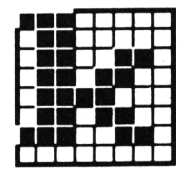
104
&H68
&X01101000



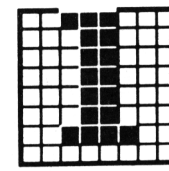
105
&H69
&X01101001



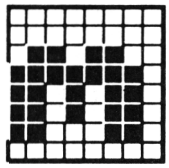
106
&H6A
&X01101010



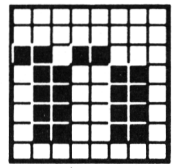
107
&H6B
&X01101011



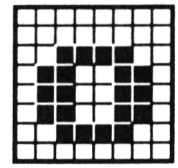
108
&H6C
&X01101100



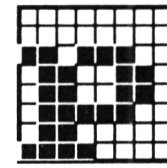
109
&H6D
&X01101101



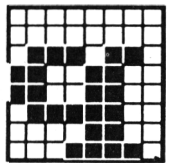
110
&H6E
&X01101110



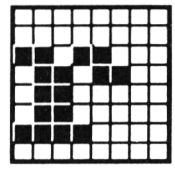
111
&H6F
&X01101111



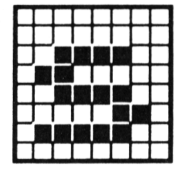
112
&H70
&X01110000



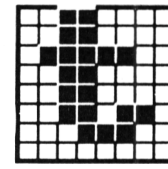
113
&H71
&X01110001



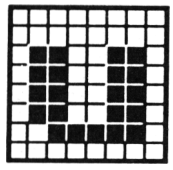
114
&H72
&X01110010



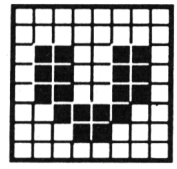
115
&H73
&X01110011



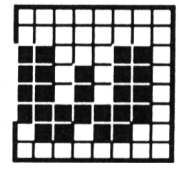
116
&H74
&X01110100



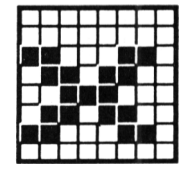
117
&H75
&X01110101



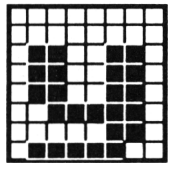
118
&H76
&X01110110



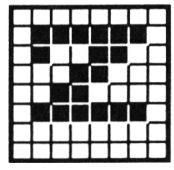
119
&H77
&X01110111



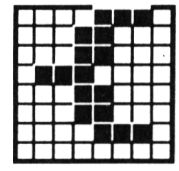
120
&H78
&X01111000



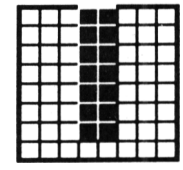
121
&H79
&X01111001



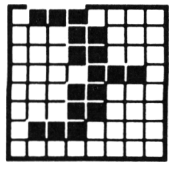
122
&H7A
&X01111010



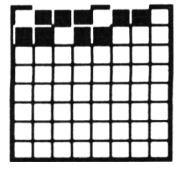
123
&H7B
&X01111011



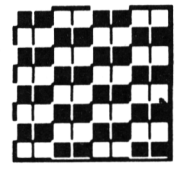
124
&H7C
&X01111100



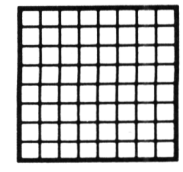
125
&H7D
&X01111101



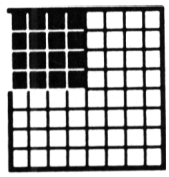
126
&H7E
&X01111110



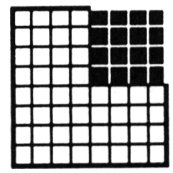
127
&H7F
&X01111111



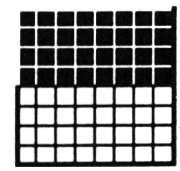
128
&H80
&X10000000



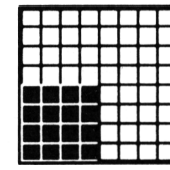
129
&H81
&X10000001



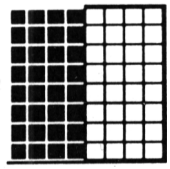
130
&H82
&X10000010



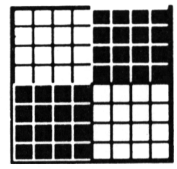
131
&H83
&X10000011



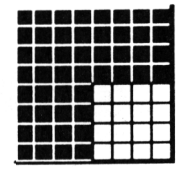
132
&H84
&X10000100



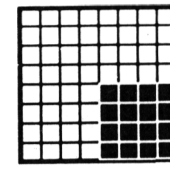
133
&H85
&X10000101



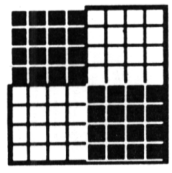
134
&H86
&X10000110



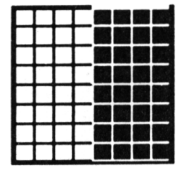
135
&H87
&X10000111



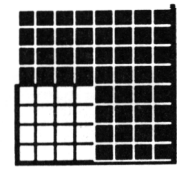
136
&H88
&X10001000



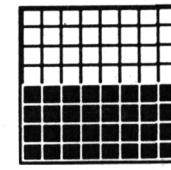
137
&H89
&X10001001



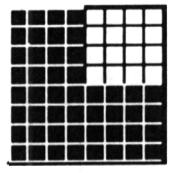
138
&H8A
&X10001010



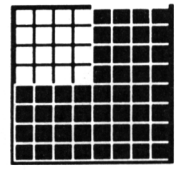
139
&H8B
&X10001011



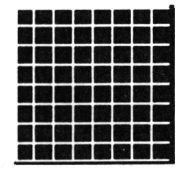
140
&H8C
&X10001100



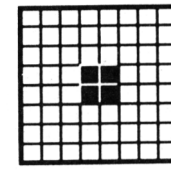
141
&H8D
&X10001101



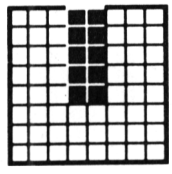
142
&H8E
&X10001110



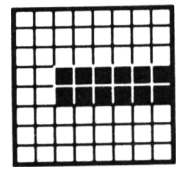
143
&H8F
&X10001111



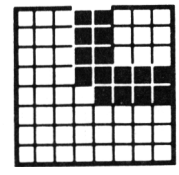
144
&H90
&X10010000



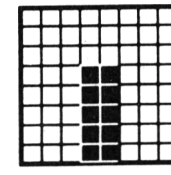
145
&H91
&X10010001



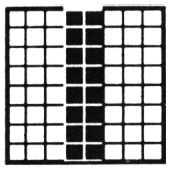
146
&H92
&X10010010



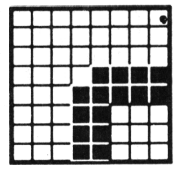
147
&H93
&X10010011



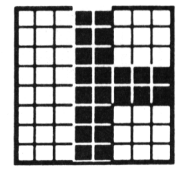
148
&H94
&X10010100



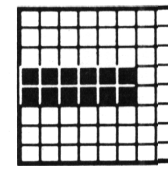
149
&H95
&X10010101



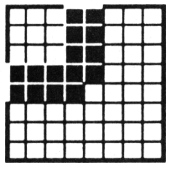
150
&H96
&X10010110



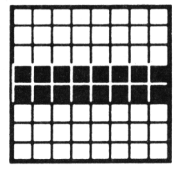
151
&H97
&X10010111



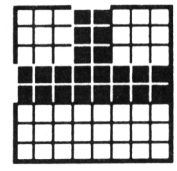
152
&H98
&X10011000



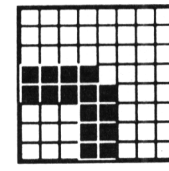
153
&H99
&X10011001



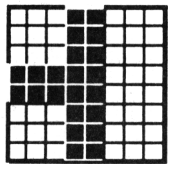
154
&H9A
&X10011010



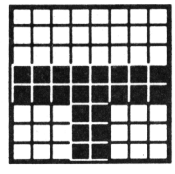
155
&H9B
&X10011011



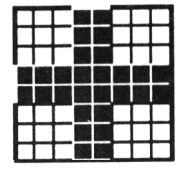
156
&H9C
&X10011100



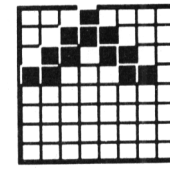
157
&H9D
&X10011101



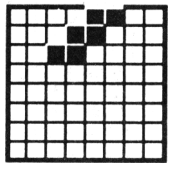
158
&H9E
&X10011110



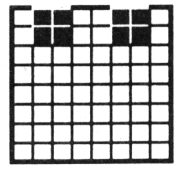
159
&H9F
&X10011111



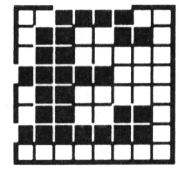
160
&HA0
&X10100000



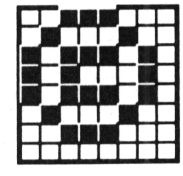
161
&HA1
&X10100001



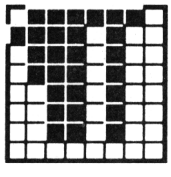
162
&HA2
&X10100010



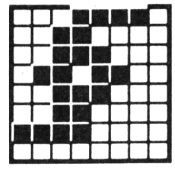
163
&HA3
&X10100011



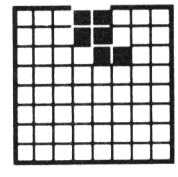
164
&HA4
&X10100100



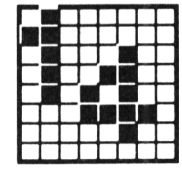
165
&HA5
&X10100101



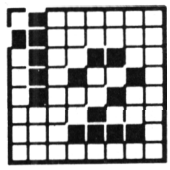
166
&HA6
&X10100110



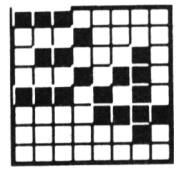
167
&HA7
&X10100111



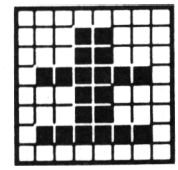
168
&HA8
&X10101000



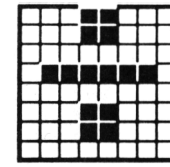
169
&HA9
&X10101001



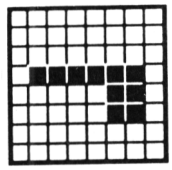
170
&HAA
&X10101010



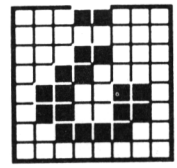
171
&HAB
&X10101011



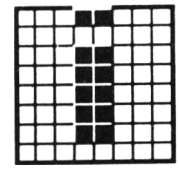
172
&HAC
&X10101100



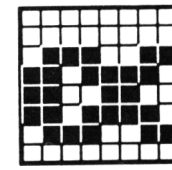
173
&HAD
&X10101101



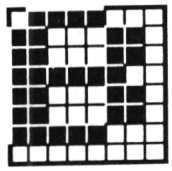
174
&HAE
&X10101110



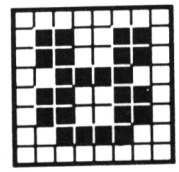
175
&HAF
&X10101111



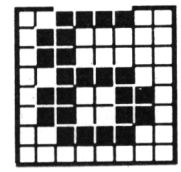
176
&HB0
&X10110000



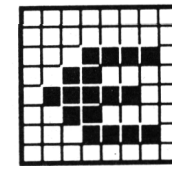
177
&HB1
&X10110001



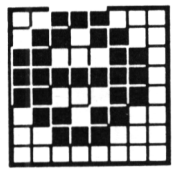
178
&HB2
&X10110010



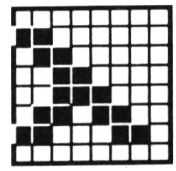
179
&HB3
&X10110011



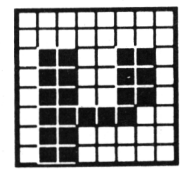
180
&HB4
&X10110100



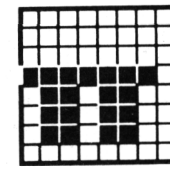
181
&HB5
&X10110101



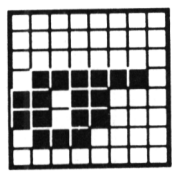
182
&HB6
&X10110110



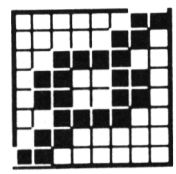
183
&HB7
&X10110111



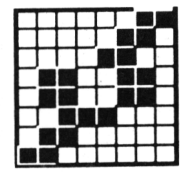
184
&HB8
&X10111000



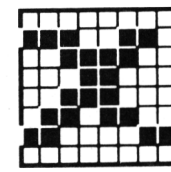
185
&HB9
&X10111001



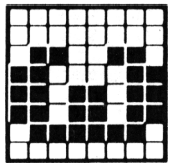
186
&HBA
&X10111010



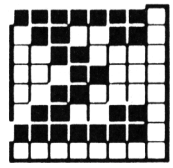
187
&HBB
&X10111011



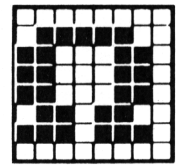
188
&HBC
&X10111100



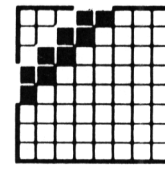
189
&HBD
&X10111101



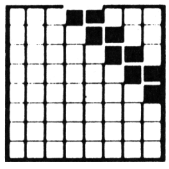
190
&HBE
&X10111110



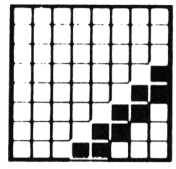
191
&HBF
&X10111111



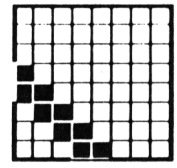
192
&HC0
&X11000000



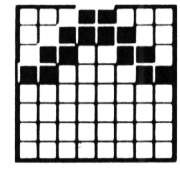
193
&HC1
&X11000001



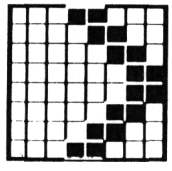
194
&HC2
&X11000010



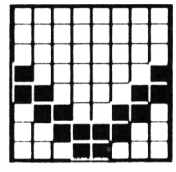
195
&HC3
&X11000011



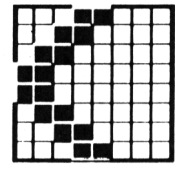
196
&HC4
&X11000100



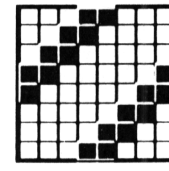
197
&HC5
&X11000101



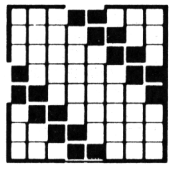
198
&HC6
&X11000110



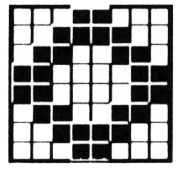
199
&HC7
&X11000111



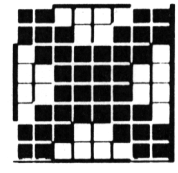
200
&HC8
&X11001000



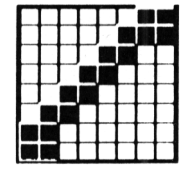
201
&HC9
&X11001001



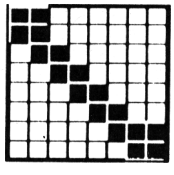
202
&HCA
&X11001010



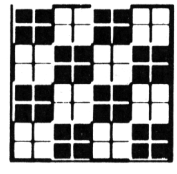
203
&HCB
&X11001011



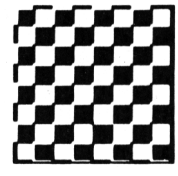
204
&HCC
&X11001100



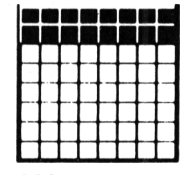
205
&HCD
&X11001101



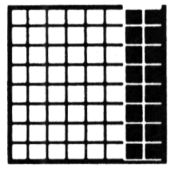
206
&HCE
&X11001110



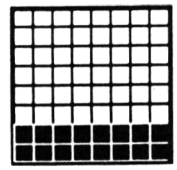
207
&HCF
&X11001111



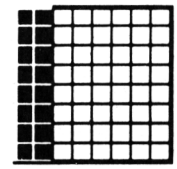
208
&HD0
&X11010000



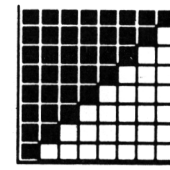
209
&HD1
&X11010001



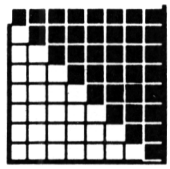
210
&HD2
&X11010010



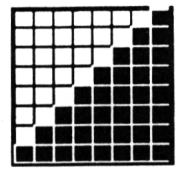
211
&HD3
&X11010011



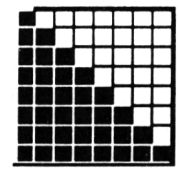
212
&HD4
&X11010100



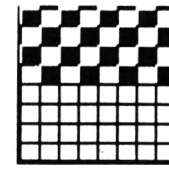
213
&HD5
&X11010101



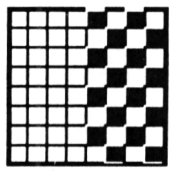
214
&HD6
&X11010110



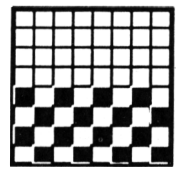
215
&HD7
&X11010111



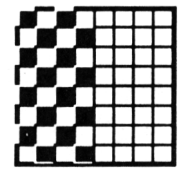
216
&HD8
&X11011000



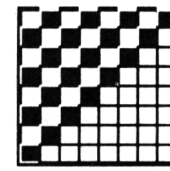
217
&HD9
&X11011001



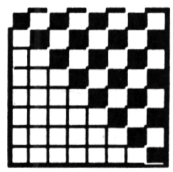
218
&HDA
&X11011010



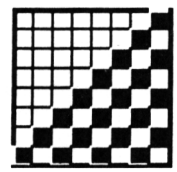
219
&HDB
&X11011011



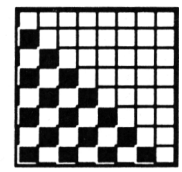
220
&HDC
&X11011100



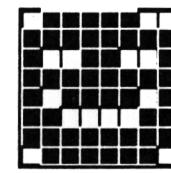
221
&HDD
&X11011101



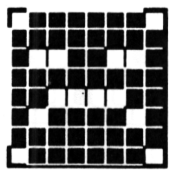
222
&HDE
&X11011110



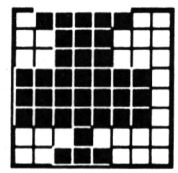
223
&HDF
&X11011111



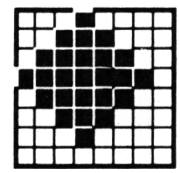
224
&HE0
&X11100000



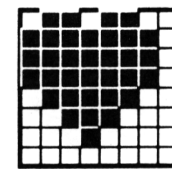
225
&HE1
&X11100001



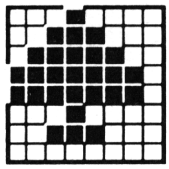
226
&HE2
&X11100010



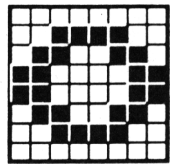
227
&HE3
&X11100011



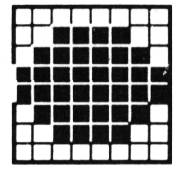
228
&HE4
&X11100100



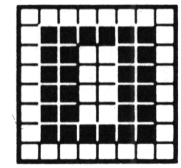
229
&HE5
&X11100101



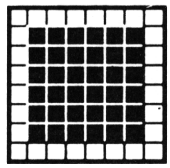
230
&HE6
&X11100110



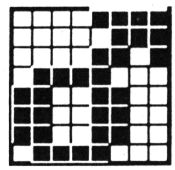
231
&HE7
&X11100111



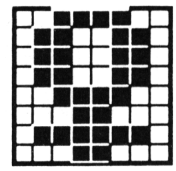
232
&HE8
&X11101000



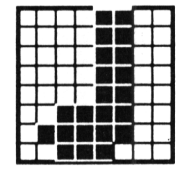
233
&HE9
&X11101001



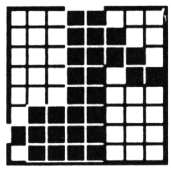
234
&HEA
&X11101010



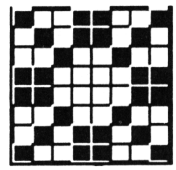
235
&HEB
&X11101011



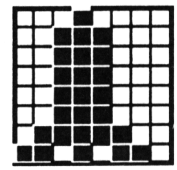
236
&HEC
&X11101100



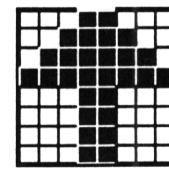
237
&HED
&X11101101



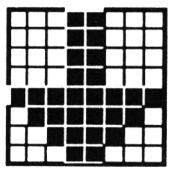
238
&HEE
&X11101110



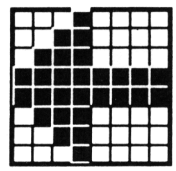
239
&HEF
&X11101111



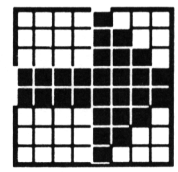
240
&HF0
&X11110000



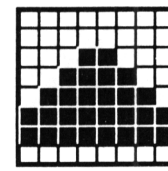
241
&HF1
&X11110001



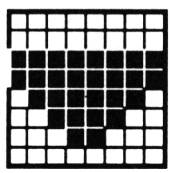
242
&HF2
&X11110010



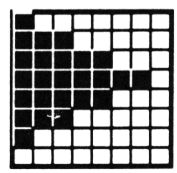
243
&HF3
&X11110011



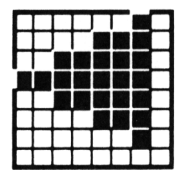
244
&HF4
&X11110100



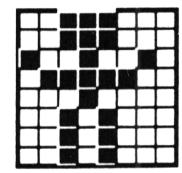
245
&HF5
&X11110101



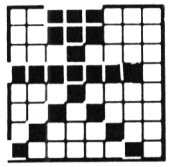
246
&HF6
&X11110110



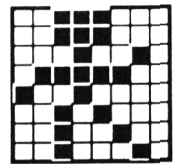
247
&HF7
&X11110111



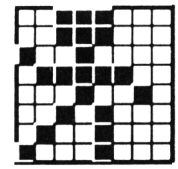
248
&HF8
&X11111000



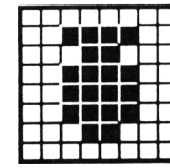
249
&HF9
&X11111001



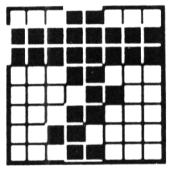
250
&HFA
&X11111010



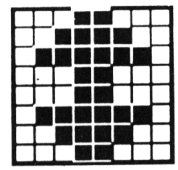
251
&HFB
&X11111011



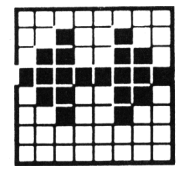
252
&HFC
&X11111100



253
&HFD
&X11111101

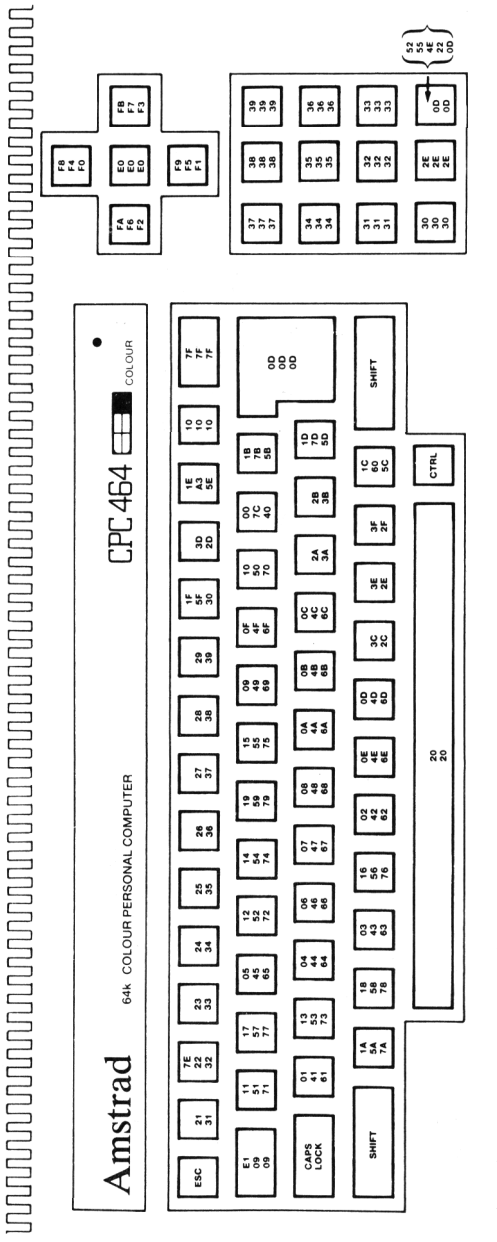


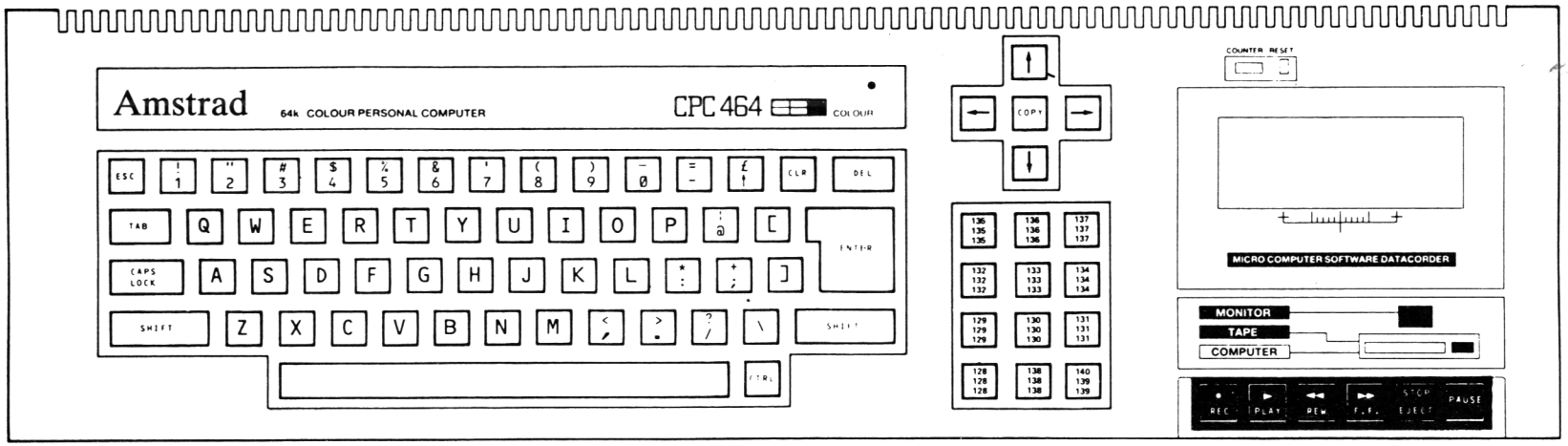
254
&HFE
&X11111110



255
&HFF
&X11111111

ASCII Standardwerte

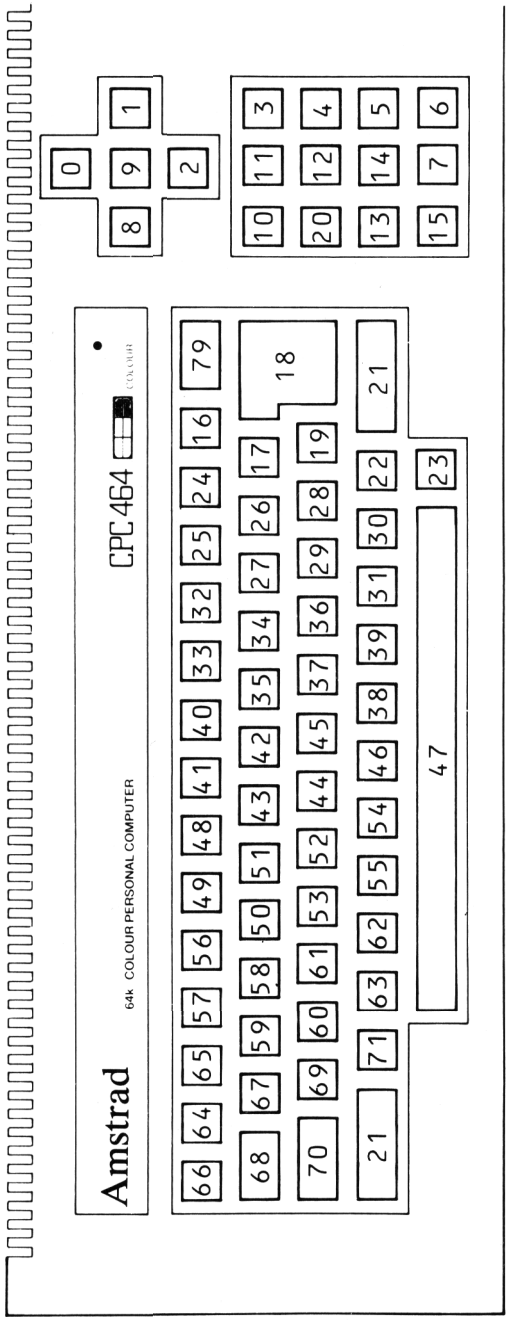




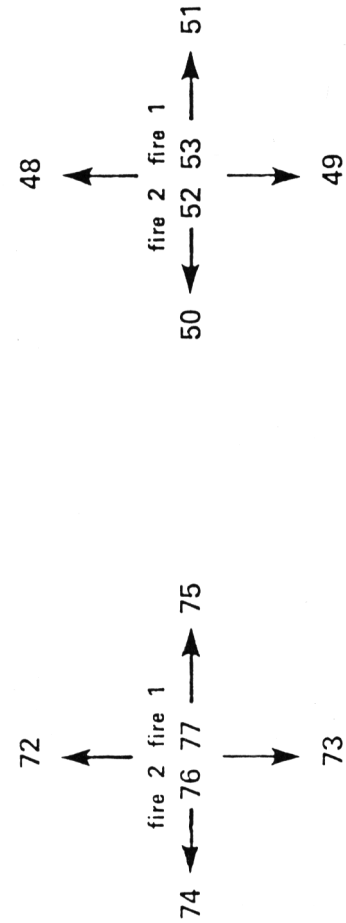
erw. Zch.	Wert	ascii
128	0	30
129	1	31
130	2	32
131	3	33
132	4	34
133	5	35
134	6	36
135	7	37
136	8	38
137	9	39
138		2E
139	[enter]	0D
140	run"	52,55,4E,22,0D

Erweiterungs-Zeichen Standard-position und Werte

Nummernzuordnung zu Tasten und Joystick-Signalen



JOYSTICK 0



JOYSTICK 1

Anhang IV: Eine Einführung für den CPC464 für Fortgeschrittene

Der CPC464 ist ein preiswerter Farb-Personal-Computer, der durch den Einsatz bester und bewährtester Techniken so entwickelt wurde, daß zu einem guten Preis-Leistungs-Verhältnis auch noch ein großer Erweiterungsspielraum geschaffen wurde. Somit ist er nicht nur für Anfänger, sondern auch für erfahrene Benutzer gleichermaßen interessant.

Die Hardware und ROM-Software wurden so konzipiert, daß sowohl dem Anfänger als auch dem erfahrenen Programmierer die Möglichkeit gegeben ist, vorhandene Software anzupassen, neue Software zu schreiben und die vielen Eigenschaften des CPC464 zu nutzen.

Die Hauptmerkmale des Systems sind:

Z80 CPU

Der am weitesten verbreitete Microprozessor, der weltweit in Heimcomputern verwendet wird, mit der am besten unterstützten Softwarebasis – besonders, da der CPC464 die Möglichkeit der CP/M-Implementierung bietet. Die einmalige Unterbrechungsbehandlung ermöglichte es, im CPC464 die völlig neuen BASIC-Anweisungen `AFTER` und `EVERY`, sowie andere ‚Echtzeit'-Funktionen zur Überwachung des Tongenerators und des Zeitgebers zu realisieren.

64K RAM

Ein großzügig bemessenes RAM ist standardmäßig eingebaut, über 42K davon stehen dem Anwender dank der ROM-Überlagerungstechnik während der BASIC-Bearbeitung direkt zur Verfügung.

Bildschirm

Der CPC464 unterstützt 3 Bildschirmmodi mit bis zu 80 Spalten Text, 27 Farben und einer Auflösung bis zu 640 x 200 Bildpunkten (pixels).

Tastatur

Eine echte ‚Schreibmaschinen'-Tastatur mit Cursorsteuertasten und einem zusätzlichen ausgelagerten Zehnerblock, der auch als Funktionstastengruppe verwendet werden kann.

Eingebauter Cassetten-Datenrecorder

Ein Datencassettenrecorder ist eingebaut, so daß der Anwender keine Probleme mit dem Anschluß, der Pegelabstimmung usw. hat. Schreiben ist entweder mit 1k-Baud oder 2k-Baud (auf Softwareebene wählbar) möglich, die Lesegeschwindigkeit wird automatisch von der Software bestimmt.

BASIC

Ein Standard-BASIC, schneller und vielseitiger als Sie es von irgendeinem BASIC erwarten, das mit Erweiterungen für Graphik und Tonausgaben sowie ausgiebiger Unterstützung der ‚Firmware' ausgestattet ist.

Erweiterter Zeichensatz

Ein 8-Bit Zeichensatz einschließlich Symbolen und Graphikzeichen, die größtenteils über die Tastatur bzw. mit der `CHR$(n)`-Funktion angesprochen werden können.

Verbrauchte Zeit

Getaktete Unterbrechungen ermöglichen Zeitmessungen.

Anwenderdefinierte Tasten

Bis zu 32 Tasten können vom Anwender mit jeweils bis zu 32 Zeichen belegt werden. Die Möglichkeit, Tasten umzudefinieren schließt auch den Wiederholungsfaktor mit ein. Ein kompletter 255-Zeichen-Satz (der gesamte ASCII-Zeichensatz und über 100 weitere) steht zur Verfügung, der wahlweise vom Anwender umdefiniert werden kann.

Unterprogramme

Viele Assembler-Unterprogramme können von BASIC aufgerufen werden.

Bildschirmmodi

Es gibt drei Bildschirmmodi:

a) Normal

Mode 1: 40 Spalten x 25 Zeilen, 4 Textfarbmöglichkeiten
320 x 200 Bildpunkte, einzeln adressierbar in 4 Farben

b) Vielfarbenmodus

Mode 0: 20 Spalten x 25 Zeilen, 16 Textfarbmöglichkeiten
160 x 200 Bildpunkte, einzeln adressierbar in 16 Farben

c) Hohe Auflösung

Mode 2: 80 Spalten x 25 Zeilen, 2 Textfarbmöglichkeiten
640 x 200 Bildpunkte, einzeln adressierbar in 2 Farben

Wahl der Farben

(Für diese Beschreibung gilt ‚schwarz', d. h. kein Aufleuchten, als Farbe.)

Egal in welchem Bildschirmmodus gearbeitet wird, der Rand kann jedes Farbenpaar annehmen, entweder in zwei Farben blinkend oder einfarbig.

Die Anzahl der zur Verfügung stehenden ‚Tinten' (INK) ist vom Bildschirmmodus abhängig. Jede Tinte kann zwei Farben (blinkend) oder nur eine (einfarbig) annehmen. Die Anzahl der benutzbaren Tinten hängt zu jeder Zeit, wie vorher definiert, vom Bildschirmmodus ab. Die Schreibfläche (PAPER), der Text-Schreibstift (text PEN) und der Graphik-Schreibstift (graphic PEN) können dann jeweils einer zur Verfügung stehenden Tinte zugeordnet werden.

Der geschriebene Text kann entweder durchscheinend oder überdeckend sein, d. h. er wird entweder die Papierfarbe belassen und nur die Graphiken überschreiben oder den Hintergrund vollständig überschreiben.

Windows – Fenster

Sie können bis zu acht Textfenster auswählen, in die die Zeichen geschrieben werden, und auch ein Graphikfenster zur Darstellung von Graphiken.

Die Fenster werden auf Stanardwerte zurückgesetzt, wenn Sie den Modus für den Bildschirm setzen.

Anmerkung: Wenn das Textfenster den ganzen Bildschirm umfaßt (Normalzustand), dann sorgt die Hardware für ein schnelles Auf- und Abrollen. Wenn ein kleineres Textfenster gewählt wurde, geschieht das Auf- und Abrollen durch die Software und ist dementsprechend langsamer.

Cursor

Der Cursor ist außer Betrieb, wenn die CPU keine Eingabe von der Tastatur erwartet; er fordert also, sobald er erscheint, automatisch zur Eingabe auf. Der Cursor wird durch ein Rechteck mit umgekehrten Farbwerten dargestellt.

Mehrstimmiger Klang

Die vielfältigen Klänge, die mit dem CPC464 erzeugt werden können, werden von einem Standard-Tongenerator aus der AY8910 Baureihe von General Instruments produziert. Das Gerät arbeitet mit drei Kanälen (Stimmen), von denen jeder unabhängig in Klang und Lautstärke gesteuert werden kann. Weißes Rauschen kann nach Belieben hinzugefügt werden.

Die drei Kanäle erscheinen links, rechts und in der Mitte (Ausgang über Stereoanschlußbuchse). Der eingebaute Lautsprecher gibt alle Kanäle in Mono aus.

Die Software bietet die Möglichkeit, die Hüllkurven von Ton und Lautstärke zu steuern. Die interne Lautstärkensteuerung des Tongenerators wird meistens nicht benutzt.

Druckeranschluß

Es ist ein standardisierter Centronics kompatibler Anschluß für einen Paralleldrucker eingebaut, bei dem das ‚Busy'-Signal für Handshake-Operationen verwendet werden kann.

Erweiterungsunterstützung

Viele Hardwareschnittstellen sind über Jumpblöcke oder andere Einrichtungen ansprechbar, um den Ausbau der Software zu ermöglichen. So werden von Amstrad u. a. Disketten-Laufwerke, serielle Schnittstellen mit Treibersoftware in ROMs usw. erhältlich sein.

Erweiterungs-ROMs

Alle ROMs belegen die oberen 16K des Speichers (wo auch das BASIC liegt), und über die Firmware ist es möglich, bis zu 240 zusätzliche 16K ROMs anzusprechen (dabei wird das Grundgerät um einen Adress-Decoder erweitert, der Teil der ROM-Erweiterungshardware ist).

Koordinaten

Der Textbeginn ist in der linken oberen Ecke das Bildschirms. Die tatsächlichen Positionen am Bildschirm sind vom Bildschirmmodus abhängig.

Graphik-Ausgaben beginnen in der linken unteren Ecke des Bildschirms. Es wird angenommen, daß der Bildschirm ständig im Hochauflösungsmodus ist – obwohl die Berechnungen für Tinten in jedem Bildschirmmodus richtig durchgeführt werden.

Beachte: Im normalen Bildschirmmodus hat jeder Bildpunkt (pixel) **zwei** waagrechte Adressen, und eine von beiden kann benutzt werden. Im Vielfarbenmodus hat jeder Bildpunkt **vier** waagrechte Adressen, und irgendeine der vier darf benutzt werden, um die Posititon zu bestimmen. Die senkrechte Achse hat Koordinaten von 0 bis 399, die durch zwei geteilt und ganzzahlig als tatsächliche Position von 0 bis 199 zur Verfügung gestellt werden. Dadurch können die Proportionen am Bildschirm besser dargestellt werden.

Überblick:

Eine kurze Zusammenfassung der wichtigsten Eigenschaften der Hardware und der Firmware des CPC464.

1) Hardware

1.1) Im Gehäuse des CPC464:

Computer, Tastatur, Cassetten-Datacorder und Lautsprecher. RGB- und Luminanz-Ausgänge.

1.1.1) LSI Chips

Z80A Prozessor mit 4 MHz Taktfrequenz.
64K Bytes mit 64K x 1 dynamischem RAM, aufgefrischt durch Zugriff auf Bildschirmspeicher.
32K Bytes ROM, mit BASIC und Betriebssystem (OS).

Eine speziell entwickelte Logik-Schaltung enthält fast alle Logik, die nicht bereits in den LSI-Chips steckt; vor allem die Zeitgeber, die Farberzeugung und die DMA-Schaltung.

Die 6845 CRT Controller-Einheit erzeugt die Abtastsignale für das Video-RAM.

Beachte:

Das Mapping ist kompliziert und ändert sich mit dem Bildschirmmodus. Der CRTC kann für ‚Scrolling' (seitlich um 1/40 Bildschirmbreite) und ‚Rolling' (Auf und Ab um 8 Abtastlinien) über die Software angesprochen werden. Über Parameter kann der CRTC die Zeilenzahl, die Rasterung und die Breite und Lage des Bildrandes steuern.

Tongenerator-Baustein AY-3-8912 von GI: 3 Stimmen. Der Ton der 3 Kanäle wird gleichmäßig gemischt und mono über den eingebauten Lautsprecher ausgegeben, wobei die Lautstärke geregelt werden kann. Es gibt auch einen externen Stereo-Ausgang, wobei gilt

Links = Kanal A + 1/2 Kanal C; Rechts = Kanal B + 1/2 Kanal C.
Dieser Chip bekommt auch die Informationen, die von der Tastatur oder vom Joystick-Anschluß kommen.

Ein 8255 paralleler I/O-Baustein bildet die Schnittstelle vom Bus zum GI-Tonbaustein. Er tastet auch die Tastatur, den Joystick-Port und zusätzliche Anschlüsse ab und steuert die Cassette.

1.2) Außerhalb des Gehäuses

Zum CPC464 gibt es zwei verschiedene direkt anzuschließende Videomonitoren. Beide haben eine 5V Stromversorgung für den Computer und sind an die Netzspannung im Verkaufsland angepaßt. Zusätzlich gibt es den **MP2**, ein kleines Gerät mit Stromversorgung und UHF-Modulator. Für den Anschluß eines Centronics-kompatiblen Druckers und einer HiFi-Anlage braucht man noch jeweils ein Verbindungskabel.

1.3) Anzeigetechnik

Der Bildschirm wird aus einem 16K Speicherbereich vesorgt. Die Maschine stellt 27 Farben zur Verfügung, von denen nach Belieben eine Palette ausgewählt werden kann. Die Anzahl der unterschiedlichen Tinten (INKs) auf der Palette hängt vom Bildschirmmodus ab. Mehreren Tinten kann, falls nötig, die gleiche Farbe zugewiesen werden. Die Pixels auf dem Schirm sind definiert als Punkte in einer bestimmten Farbe.
(Beachten Sie, daß der Hintergrund, oder das Papier `(PAPER)` eine Farbe aus der vorhandenen Palette belegt.) Die beschreibbare Bildfläche wird von einem Rand `(BORDER)` umgeben, der völlig unabhängig eine der 27 Farben haben kann.

Modus	Anz. Tinten	Senkr. Punkte	Waagr. Punkte	Waagr. Zeichen
Normal	4	200	320	40
Hochauflösung	2	200	640	80
Vielfarbig	16	200	160	20

Auf einem einfarbigen Monitor produziert der Computer verschiedene Grautöne. Ihre Reihenfolge mit zunehmender Helligkeit ist folgendermaßen:

Grauton	Farbe	Grauton	Farbe
0	Schwarz	13	Weiß
1	Blau	14	Pastellblau
2	Hellblau	15	Orange
3	Rot	16	Rosa
4	Magenta	17	Pastellmagenta
5	Hellviolett	18	Hellgrün
6	Hellrot	19	Seegrün
7	Purpur	20	Helles Blaugrün
8	Helles Magenta	21	Limonengrün
9	Grün	22	Pastellgrün
10	Blaugrün	23	Pastellblaugrün
11	Himmelblau	24	Hellgelb
12	Gelb	25	Pastellgelb
		26	Leuchtendweiß

1.4) Speicherbelegung

Die 64K des RAM sind wie folgt aufgeteilt:

Beachten Sie, daß ein Teil des ROM das Screen-RAM (Bildschirm) überlagert und dabei den größtmöglichen Bereich für das Benutzer-RAM während BASIC-Operationen freiläßt.

	0000H	
ROM SECTION 0 <		
	3FFFH	
	4000H	
	7FFFH	
	8000H	
	BFFFH	
	C000H	
ROM SECTION 1 <		>RAM [Screen 3]
	FFFFH	

Wenn sowohl RAM als auch ROM auf der gleichen Adresse liegen, dann wird beim Lesen das ROM und beim Schreiben das RAM angesprochen. Jeder der beiden ROM-Bereiche kann abgeschaltet werden, um auf der gleichen Adresse einen Lesezugriff auf das RAM zu ermöglichen.

1.5) Zusätze

1.5.1) ROMs

Zusätzliche ROMs können an Stelle irgendeines Teils des eingebauten ROMs eingesetzt werden. Die Adressen-Auswahl und die ‚Bank Selection'-Logik sind in einem Modul, das an den Erweiterungsbus angeschlossen wird, untergebracht. Alle Signale, die dort benötigt werden, liegen am Erweitertungsbus an.

1.5.2) RAM-Erweiterung

Ein zusätzlicher RAM-Bereich kann an Stelle irgendeines Teiles des eingebauten RAMs zugeschaltet werden. Die Adressen-Auswahl und die ‚Bank-Selection'-Logik sind in einem Modul, das an den Erweiterungsbus angeschlossen wird, untergebracht. Alle Signale, die dort benötigt werden, liegen am Erweiterungsbus an. Dieser Speicherbereich kann nur gelesen werden. Um I/O-Operationen durchführen zu können, ist noch ein spezieller Zusatz erforderlich, der es dem Computer erlaubt, dieses zusätzliche RAM zu beschreiben.

1.5.3) Zusätzliche Ein-/Ausgaben (I/O)

Die meisten Adressen für I/O-Anschlüsse sind vom Computer belegt, insbesondere sollten die Adressen unterhalb von 7Fxx auf keinen Fall benützt werden. Die folgenden Adressen können von externer Hardware verwendet werden:

F8xx, F9xx, FAxx, FBxx

Perphere Geräte am Erweiterungsbus müssen die Adressen A0 bis A7 decodieren; Adresse A10 ist low. Die I/O Kanäle am Erweiterungsbus im Adreßbereich F8xx bis FBFF sind wie folgt reserviert:

Adressen A0-A7

00	-	7B	**nicht benützen**
7C	-	7F	reserviert für Disk-Schnittstelle
80	-	BB	**nicht benützen**
BC	-	BF	reserviert für zukünftige Verwendung
C0	-	DB	**nicht benützen**
DC	-	DF	reserviert für Kommunikations-Schnittstellen
E0	-	FF	stehen dem Anwender für periphere Geräte zur Verfügung

Beachten Sie, daß Z80 Befehle, die das B Register auf die obere Hälfte des Adressenbus (A15-A8) setzen, verwendet werden müssen.

2) Tastatur

Ein vollständiges Zurücksetzen des Computers wird durch gleichzeitiges Drücken von [CRTL] [SHIFT] [ESC] erreicht. Tasten, die abdruckbare Zeichen oder Cursorbewegungen erzeugen, haben eine über die Firmware gesteuerte Wiederholfunktion, ausgenommen alle Tasten im separaten Ziffernblock.

[ESC] unterbricht die Ausführung eines Programms. Wird ein zweites [ESC] gegeben, wird die Ausführung beendet. Drückt man jedoch irgendeine andere Taste, wird die Ausführung fortgesetzt.

[CAPS LOCK] ist eine Umschaltfunktion, die über die caps lock Taste gesteuert wird. Shift lock ist eine Umschaltfunktion, die bestätigt wird, wenn man [CTRL] [CAPS LOCK] gleichzeitig drückt.

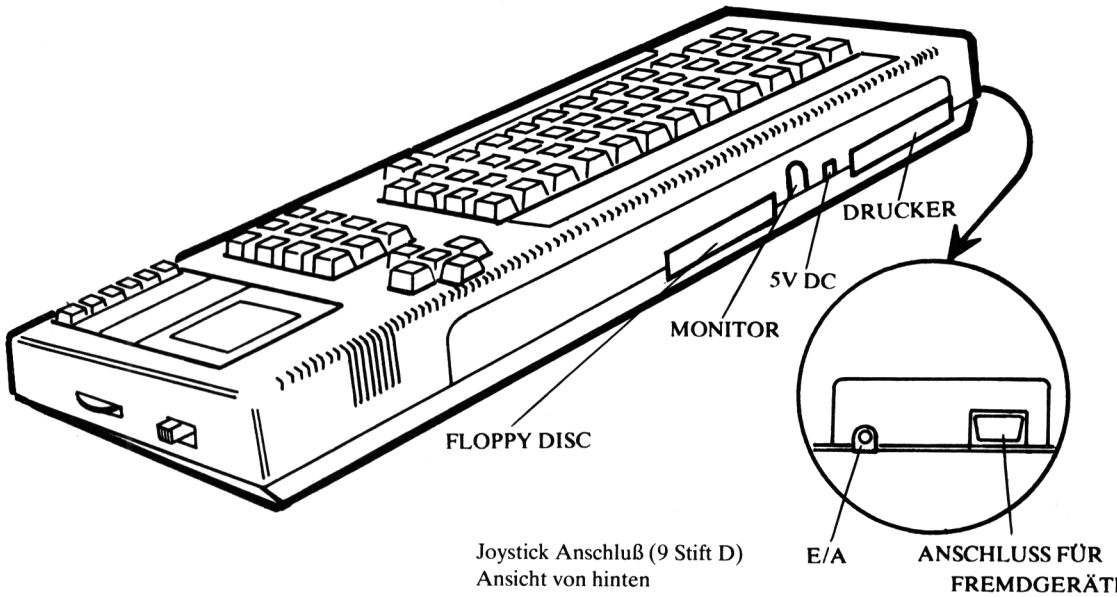
Der Copy Cursor wird vom Eingabecursor getrennt, wenn man zusammen mit einer Cursortaste die [SHIFT] Taste drückt. Man bekommt das Zeichen, über dem der Copy Cursor steht, in die Eingabezeile, wenn man die [COPY] Taste drückt.

Die Cursortasten erlauben ein Editieren des Eingabepuffers, der sich über mehrere Bildschirmzeilen erstrecken kann. Mit diesen Tasten kann der Anfang der Tastatureingabe auf jede Bildschirmposition gebracht werden, bevor eine Eingabe über die Tasten erfolgt. Sobald Tastatureingabe empfangen wird, ist die Bildschirmposition fixiert. Der neue Eingabetext überschreibt einen vorhandenen Inhalt an dieser Bildschirmposition.

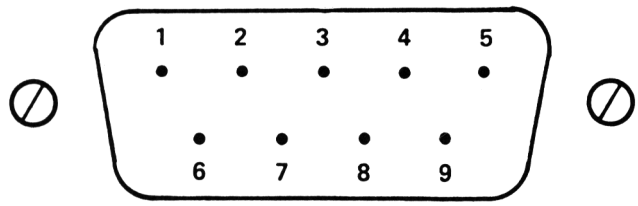
[DEL] löscht rückwärts und [CLR] löscht vorwärts.

Anhang V:

Anschlüsse auf der Rückseite vom CPC464

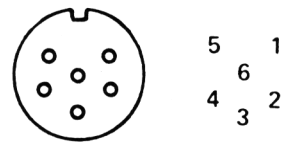


Joystick Anschluß (9 Stift D)
Ansicht von hinten



Stift 1 oben Stift 2 unten Stift 3 links Stift 4 rechts Stift 5 frei	Stift 6 Feuer 2 Stift 7 Feuer 1 Stift 8 COMMON Stift 9 COM 2

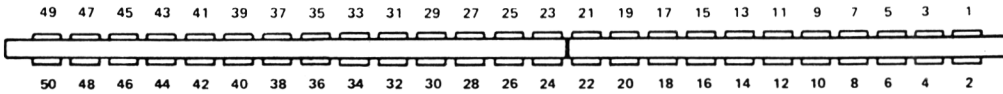
VIDEO AUSGANG (6 Stift DIN)
Geräusch Generator



Stift 1 rot Stift 2 grün Stift 3 blau	Stift 4 Sync Stift 5 Erde Stift 6 Leuchte

Erweiterungsanschluß 50 polige 0,1 Steckliste

Ansicht von hinten

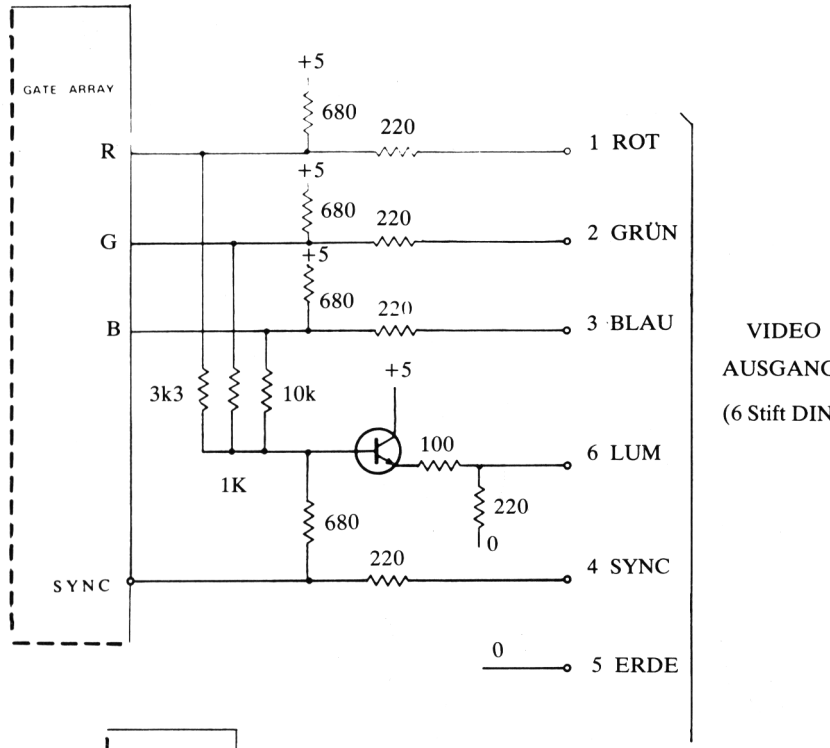


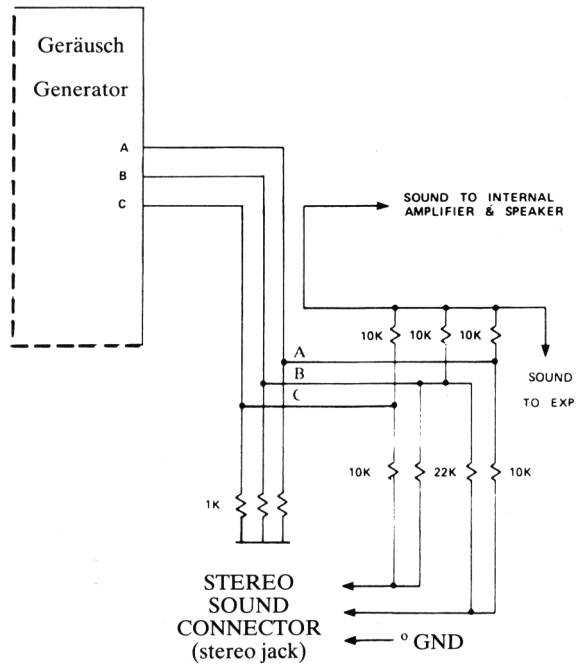
PIN	1	SOUND	PIN	18	A0	PIN	35	$\overline{\text{INT}}$
PIN	2	GND	PIN	19	D7	PIN	36	$\overline{\text{NMI}}$
PIN	3	A15	PIN	20	D6	PIN	37	$\overline{\text{BUSRD}}$
PIN	4	A14	PIN	21	D5	PIN	38	$\overline{\text{BUSAK}}$
PIN	5	A13	PIN	22	D4	PIN	39	READY
PIN	6	A12	PIN	23	D3	PIN	40	$\overline{\text{BUS RESET}}$
PIN	7	A11	PIN	24	D2	PIN	41	$\overline{\text{RESET}}$
PIN	8	A10	PIN	25	D1	PIN	42	$\overline{\text{ROMEN}}$
PIN	9	A9	PIN	26	D0	PIN	43	$\overline{\text{ROMDIS}}$
PIN	10	A8	PIN	27	+ 5v	PIN	44	$\overline{\text{RAMRD}}$
PIN	11	A7	PIN	28	$\overline{\text{MREQ}}$	PIN	45	RAMDIS
PIN	12	A6	PIN	29	$\overline{\text{M1}}$	PIN	46	CURSOR
PIN	13	A5	PIN	30	$\overline{\text{RFSH}}$	PIN	47	L. PEN
PIN	14	A4	PIN	31	$\overline{\text{IORQ}}$	PIN	48	$\overline{\text{EXP}}$
PIN	15	A3	PIN	32	$\overline{\text{RD}}$	PIN	49	GND
PIN	16	A2	PIN	33	$\overline{\text{WR}}$	PIN	50	ϕ
PIN	17	A1	PIN	34	$\overline{\text{HALT}}$			

Druckeranschluß 34 polige 0,1 Steckleiste

Ansicht von hinten

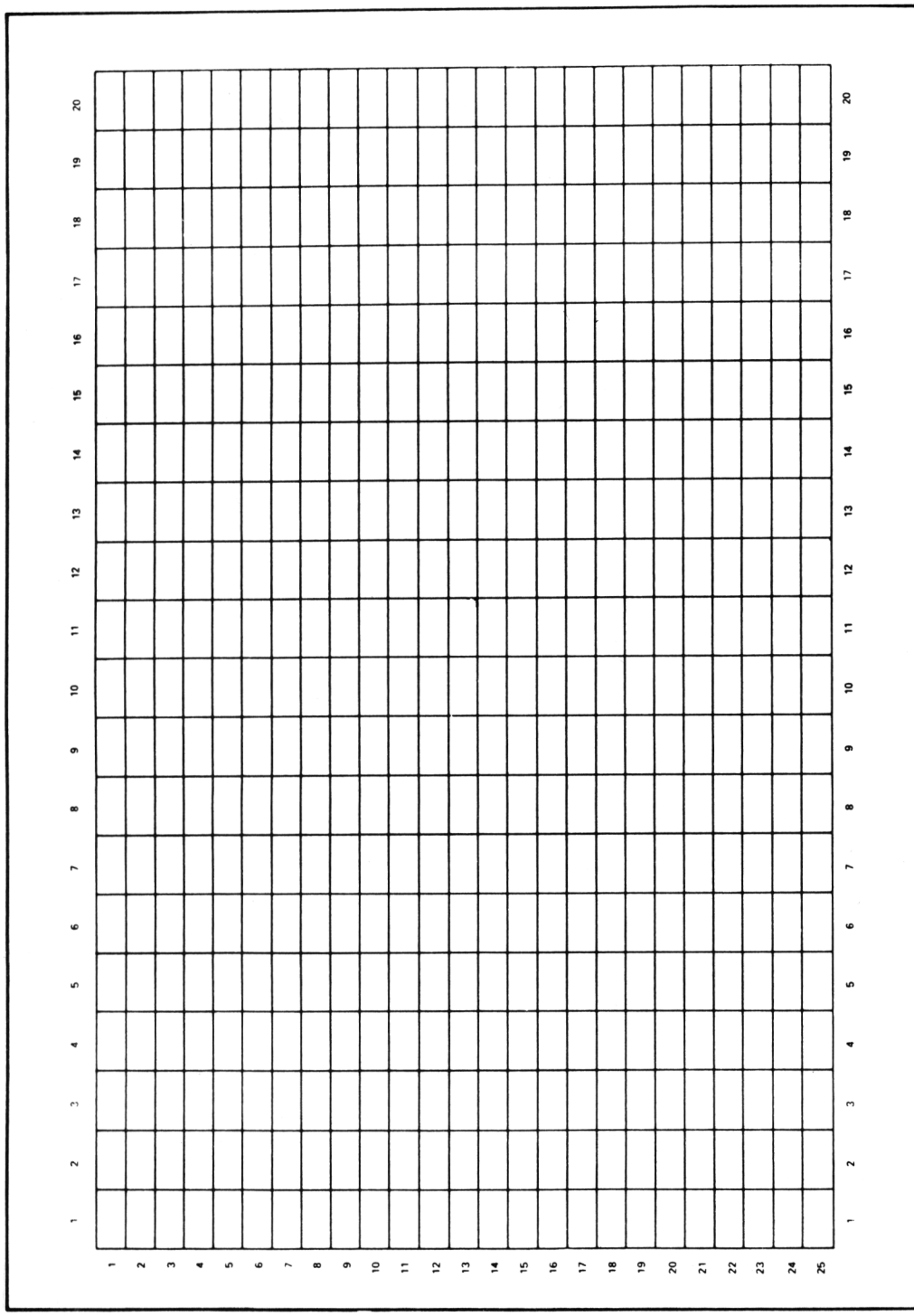
PIN	1	$\overline{\text{STROBE}}$	PIN	19	GND
PIN	2	D0	PIN	20	GND
PIN	3	D1	PIN	21	GND
PIN	4	D2	PIN	22	GND
PIN	5	D3	PIN	23	GND
PIN	6	D4	PIN	24	GND
PIN	7	D5	PIN	25	GND
PIN	8	D6	PIN	26	GND
PIN	9	GND	PIN	28	GND
PIN	11	BUSY	PIN	33	GND
PIN	14	GND			
PIN	16	GND	All other pins NC		


GATE ARRAY
R
G
B
SYNC
+5
680
220
1 ROT
2 GRÜN
3 BLAU
3k3
10k
1K
100
6 LUM
220
0
4 SYNC
5 ERDE
VIDEO
AUSGANG
(6 Stift DIN)

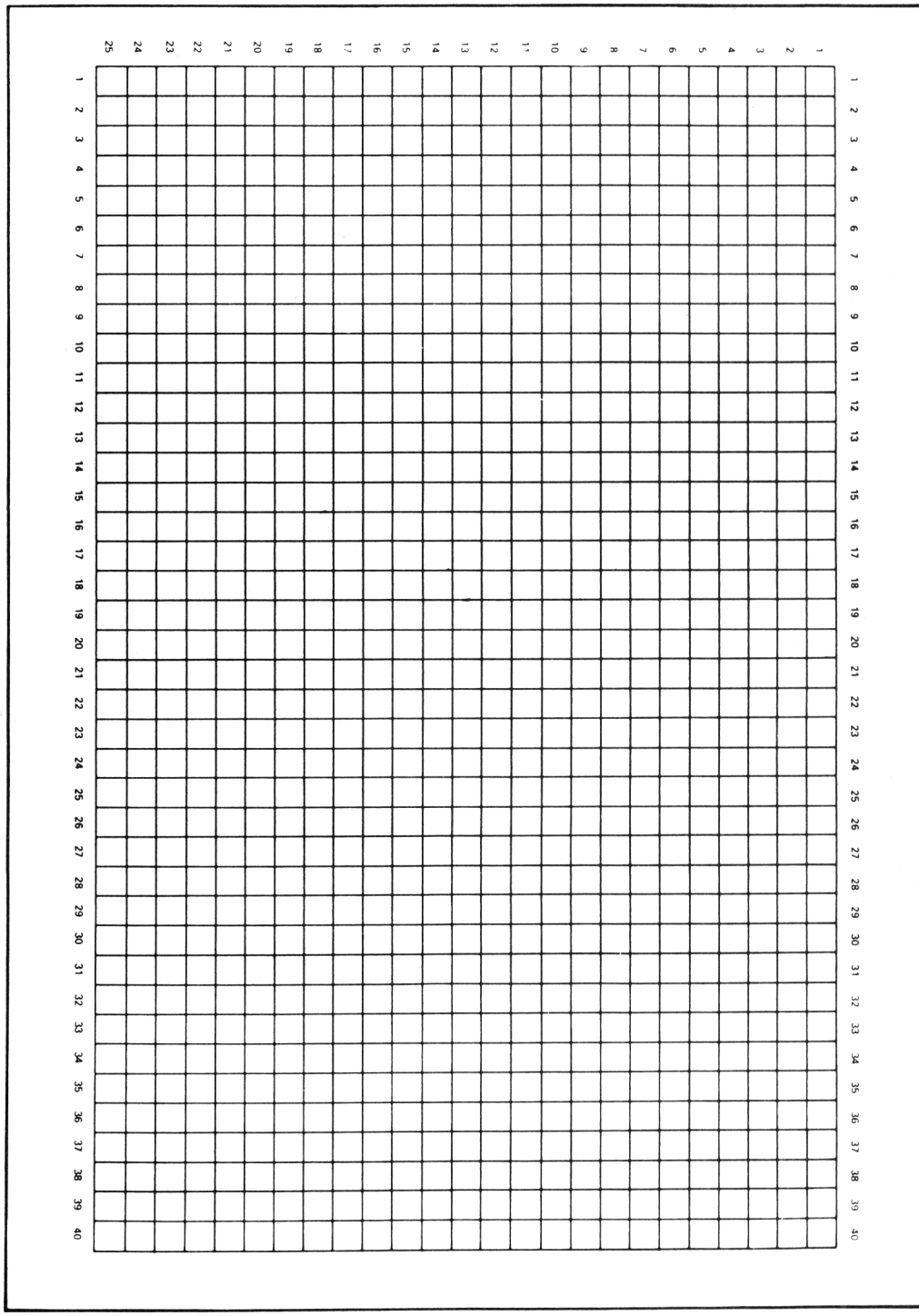

Geräusch
Generator
A
B
C
SOUND TO INTERNAL
AMPLIFIER & SPEAKER
10K
22K
1K
SOUND
TO EXP
STEREO
SOUND
CONNECTOR
(stereo jack)
GND

Anhang VI:

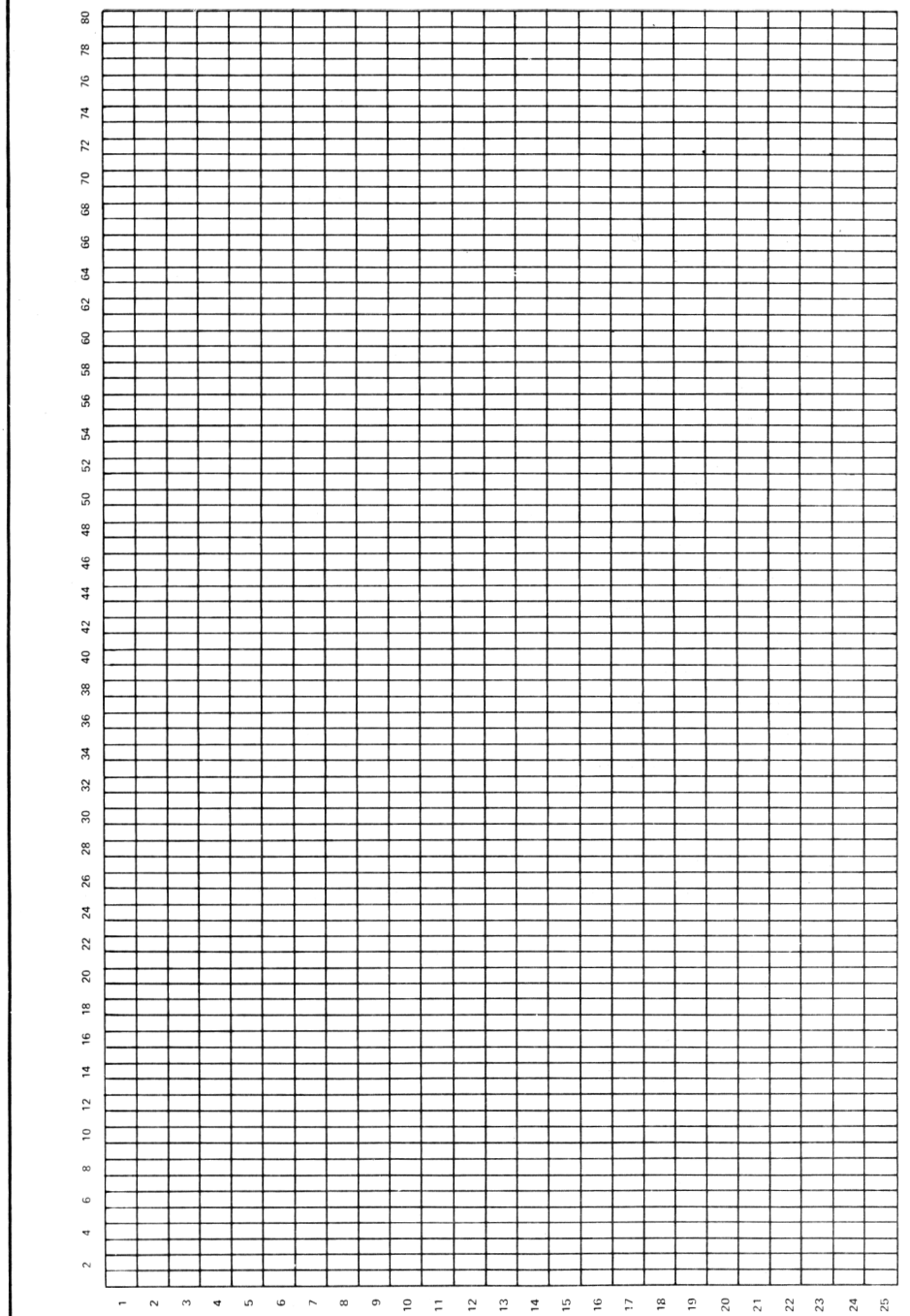
TEXT und WINDOW Planer
MODE 0 20 Spalten



TEXT und WINDOW Planer
MODE 1 40 Spalten



TEXT und WINDOW Planer
MODE 2 80 Spalten



ANHANG VII: Noten und Tonperioden

Die folgende Tabelle beschreibt die empfohlenen Tonperioden für die Noten aller acht Oktaven.

Die wirklich erzeugte Frequenz ist nicht die genau zutreffende Frequenz, da die Periodenangabe ganzzahlig sein muß. Der relative Fehler ist das Verhältnis der Differenz von genauer und erzeugter Frequenzen zur genauen Frequenz, d. h. `(GENAUE - ERZEUGTE)/GENAUE)`.

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	32.703	3822	-0.007%	
C#	34.648	3608	+0.007%	
D	36.708	3405	-0.007%	
D#	38.891	3214	-0.004%	
E	41.203	3034	+0.009%	
F	43.654	2863	-0.016%	Octave -2
F#	46.249	2703	+0.009%	
G	48.999	2551	-0.002%	
G#	51.913	2408	+0.005%	
A	55.000	2273	+0.012%	
A#	58.270	2145	-0.008%	
B	61.735	2025	+0.011%	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	65.406	1911	-0.007%	
C#	69.296	1804	+0.007%	
D	73.416	1703	+0.022%	
D#	77.782	1607	-0.004%	
E	82.407	1517	+0.009%	
F	87.307	1432	+0.019%	Octave -3
F#	92.499	1351	-0.028%	
G	97.999	1276	+0.037%	
G#	103.826	1204	+0.005%	
A	110.000	1136	-0.032%	
A#	116.541	1073	+0.039%	
B	123.471	1012	-0.038%	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	130.813	956	+0.046%	
C#	138.591	902	+0.007%	
D	146.832	851	-0.037%	
D#	155.564	804	+0.058%	
E	164.814	758	-0.057%	Octave -1
F	174.614	716	+0.019%	
F#	184.997	676	+0.046%	
G	195.998	638	+0.037%	
G#	207.652	602	+0.005%	
A	220.000	568	-0.032%	
A#	233.082	536	-0.055%	
B	246.942	506	-0.038%	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	261.626	478	+0.046%	Middle C
C#	277.183	451	+0.007%	
D	293.665	426	+0.081%	
D#	311.127	402	+0.058%	
E	329.628	379	-0.057%	
F	349.228	358	+0.019%	Octave 0
F#	369.994	338	+0.046%	
G	391.995	319	+0.037%	
G#	415.305	301	+0.005%	
A	440.000	284	-0.032%	International A
A#	466.164	268	-0.055%	
B	493.883	253	-0.038%	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	523.251	239	+0.046%	
C#	554.365	225	-0.215%	
D	587.330	213	+0.081%	
D#	622.254	201	+0.058%	
E	659.255	190	+0.206%	
F	698.457	179	+0.019%	Octave 1
F#	739.989	169	+0.046%	
G	783.991	159	-0.277%	
G#	830.609	150	-0.328%	
A	880.000	142	-0.032%	
A#	932.328	134	-0.055%	
B	987.767	127	+0.356%	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	1046.502	119	-0.374%	
C#	1108.731	113	+0.229%	
D	1174.659	106	-0.390%	
D#	1244.508	100	-0.441%	
E	1318.510	95	+0.206%	
F	1396.913	89	-0.543%	Octave 2
F#	1479.978	84	-0.548%	
G	1567.982	80	+0.350%	
G#	1661.219	75	-0.328%	
A	1760.000	71	-0.032%	
A#	1864.655	67	-0.055%	
B	1975.533	63	-0.435%	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	2093.004	60	+0.462%	
C#	2217.461	56	-0.662%	
D	2349.318	53	-0.390%	
D#	2489.016	50	-0.441%	
E	2637.021	47	-0.855%	
F	2793.826	45	+0.574%	Octave 3
F#	2959.955	42	-0.548%	
G	3135.963	40	+0.350%	
G#	3322.438	38	+0.992%	
A	3520.000	36	+1.357%	
A#	3729.310	34	+1.417%	
B	3951.066	32	+1.134%	

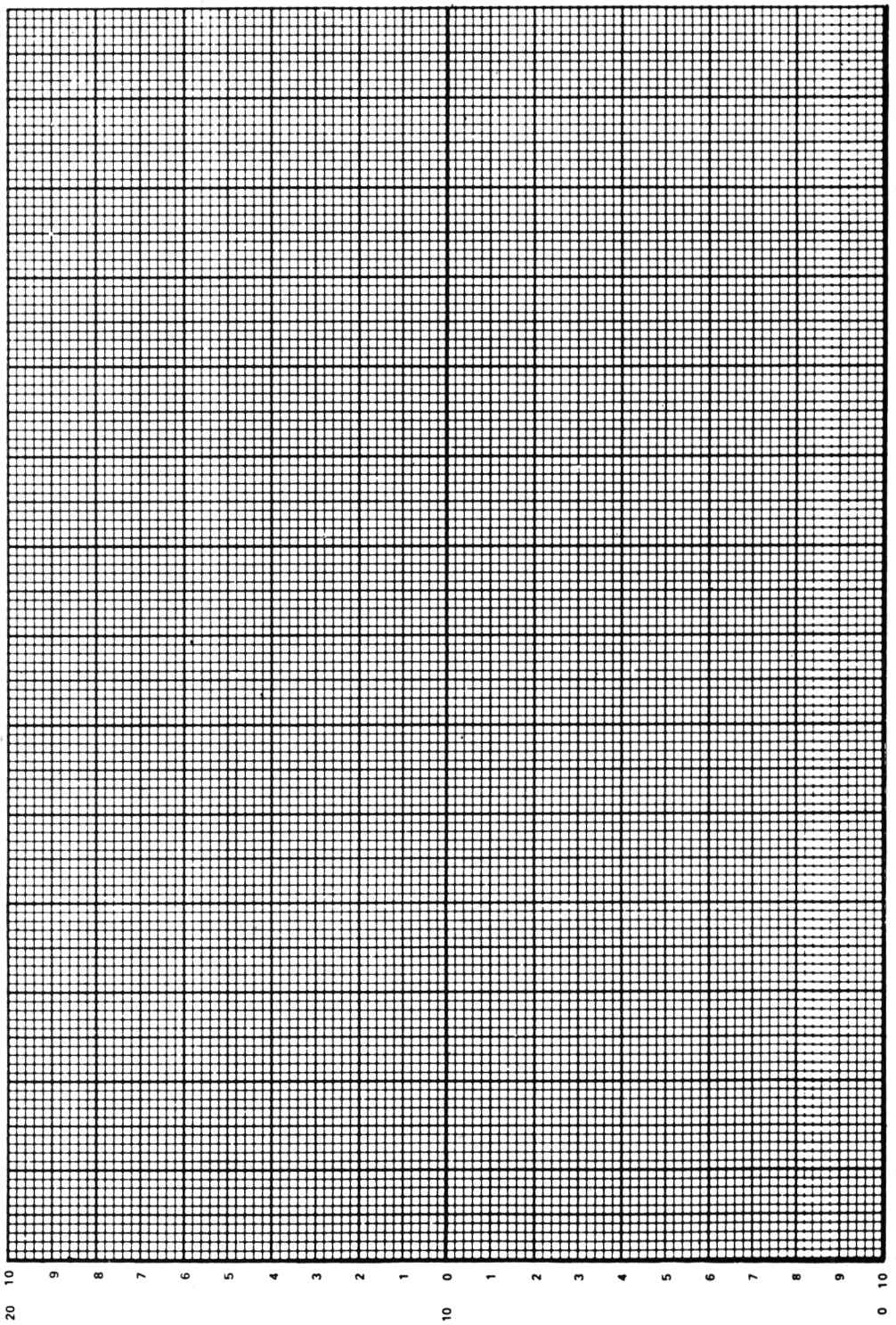
NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	4186.009	30	+0.462%	
C#	4434.922	28	-0.662%	
D	4698.636	27	+1.469%	
D#	4978.032	25	-0.441%	
E	5274.041	24	+1.246%	
F	5587.652	22	-1.685%	Octave 4
F#	5919.911	21	-0.548%	
G	6271.927	20	+0.350%	
G#	6644.875	19	+0.992%	
A	7040.000	18	+1.357%	
A#	7458.621	17	+1.417%	
B	7902.133	16	+1.134%	

Diese Werte wurden vom Internationalen A aus wie folgt errechnet:

```
FREQULENZ = 440 * (2↑(OKTAVE + (10-N)/12))
PERIODE = ROUND (125000 / FREQUENZ)
```

mit N gleich 1 für C, 2 für C#, 3 für D, usw.

Tonvariations- und Musik-Planer



Anhang VIII
Fehlercodes und reservierte Wörter

Fehlernummern und Fehlermeldungen

Wenn BASIC auf eine Programmanweisung, ein Wort oder eine Variable trifft, das es weder verstehen noch bearbeiten kann, stoppt BASIC und gibt eine Fehlermeldung aus. Die Fehlermeldung beschreibt die Fehlerursache. Falls es sich nur um einen Tippfehler handelt, schaltet BASIC in den Edit-Modus und gibt die zu korrigierende Zeile aus.

Der häufigste Fehler, den ungenaues Eintippen verursacht, ist der `Syntax Error` (Nummer 2). Falls der Fehler im Programmablauf eingetreten ist, gibt BASIC als Folge die Zeile neu aus, so daß sie sofort korrigiert werden kann. Im direkten Eingabemodus wird nur die Fehlermeldung eingegeben, da anzunehmen ist, daß die verursachende Zeile noch sichtbar ist.

Wenn das `ON ERROR GOTO`-Kommando am Programmanfang eingegeben wurde, wird beim Auftreten eines Fehlers zur angegebenen Zeile gesprungen. Im folgenden Beispiel wird zur Zeile 1000 gesprungen, falls ein Fehler entdeckt wird:

```
10    ON ERROR GOTO 1000
          (Programm)
1000 PRINT CHR$(7):MODE 2:INK 1,0:CLS : LIST
```

worauf der CPC464 piepst, den Bildschirm löscht, auf eine passende Farbkombination für den 80-spaltigen Ausdruck schaltet und das Programm auflistet, um es untersuchen zu können. Falls ein `Syntax Error` die Fehlerursache war, erscheint die zu korrigierende Zeile am Ende der Liste, und BASIC wartet im Edit-Modus auf eine Korrektur, obwohl die Meldung unterdrückt wurde.

Vergessen Sie nicht, das Programm vor Zeile 1000 zu beenden (`END`), wenn Sie Ihre Ausgaben auf dem Bildschirm behalten wollen.

BASIC erzeugt für gültige Eingaben naheliegenderweise keine Fehlermeldungen, so daß, falls irgendwann doch ein Fehler auftritt, dieser auf einen Programmierfehler zurückzuführen ist, wobei die Fehlermeldung meist hilft, die Ursache aufzuspüren. Meist lernt man ja aus seinen Fehlern, also sollten Sie das Beste aus der Tatsache machen, daß der CPC464 ein geduldiger Lehrer ist: Sie geben sicher früher auf, als der CPC464 die Geduld verliert.

Alle Fehler, die BASIC erzeugen kann, sind hier nach ihren Fehlernummern aufgelistet. Sowohl die Fehlermeldung als auch jeweils eine kurze Erklärung der möglichen Ursache sind angegeben.

1 `Unexpected NEXT` – unerwartetes NEXT

Ein `NEXT`-Kommando wurde gefunden, obwohl keine `FOR`-Schleife vorhanden war, oder die Steuervariable im `NEXT`-Kommando paßt nicht zu der im `FOR`.

2 `Syntax Error` – Syntaxfehler

BASIC versteht die angegebene Zeile nicht, da irgendetwas in ihr unzulässig ist.

3 `Unexpected RETURN` – unerwartetes RETURN

Ein `RETURN`-Kommando wurde außerhalb eines Unterprogramms gefunden.

4 `DATA exhausted` – Daten zu Ende

Ein `READ`-Kommando wurde gegeben, obwohl in `DATA`-Anweisungen keine Daten mehr vorhanden sind.

5 `Improper argument` – ungültiges Argument

Dies ist eine Mehrzweckmeldung. Der Wert eines Arguments einer Funktion oder ein Kommandoparameter ist falsch.

6 `Overflow` – Überlauf

Das Ergebnis einer arithmetischen Operation ist 'übergelaufen'. Es kann ein Gleitkommawertüberlauf sein, weil das Ergebnis größer als `1.7E-38` ist; oder es wurde versucht, eine Gleitkommazahl in eine 16 Bit große Ganzzahl mit Vorzeichen umzuwandeln.

7 `Memory full` – Speicher voll

Das Programm oder seine Variablen sind einfach zu voll, oder das Programm ist zu sehr verschachtelt (geschachtelte `GOSUB`s, `WHILE`s und `FOR`s).
Ein `Memory`-Kommando kann diese Meldung auslösen, falls der BASIC zur Verfügung gestellte Speicherplatz zu klein ist oder viel zu groß gewählt wurde. Beachten sie, daß eine geöffnete Cassettendatei Pufferplatz belegt, so daß der verfügbare Speicherplatz kleiner wird.

8 `Line does not exist` – Zeile nicht vorhanden

Die angesprochene Zeile kann nicht gefunden werden.

9 `Subscript out of range` – Subskript außerhalb des erlaubten Bereichs

Ein Subskript (Index) in einer Matrix ist zu groß oder zu klein.

10 `Array already dimensioned` – Matrix schon definiert

Eine der Matrizen eines `DIM`-Kommandos wurde schon definiert.

11 `Division by zero` – Division durch Null

Kann bei: Reeller oder Ganzzahliger Division, Divisionsreste-Bestimmung oder Potenzierung auftreten.

12 `Invalid direct command` – Falsches Direktkommando

Das zuletzt versuchte Kommando ist im direkten Modus nicht erlaubt.

13 `Type mismatch` – Typenunstimmigkeit

Ein numerischer Wert wurde angegeben, obwohl ein Textzeichen verlangt wurde oder umgekehrt, oder es wurde in einem `READ` oder `INPUT` eine Zahl des falschen Typs eingegeben.

14 `String space full` – Textzeichenbereich voll

Es wurden zuviele Zeichenketten erstellt, so daß selbst nach dem Auffüllen aller freien Speicherplätze im Programm kein Platz mehr ist.

15 `String too long` – Zeichenkette zu lang

Eine Zeichenkette hat mehr als 255 Zeichen. Kann auftreten, wenn mehrere Zeichenketten aneinandergehängt werden (addiert).

16 `String expression too complex` – Zeichenkettenausdruck zu komplex

Zeichenkettenausdrücke können viele Zwischenketten erzeugen. Wenn deren Anzahl eine vernünftige Grenze überschreitet, gibt BASIC mit dieser Meldung auf.

17 `Cannot CONTinue` – `CONT` funktioniert nicht

Das aktuelle Programm kann aus irgendeinem Grund mit `CONT` nicht wieder gestartet werden. Beachten Sie, daß `CONT` nur für das Wiederstarten eines Programms nach einem `STOP`-Kommando, `[ESC][ESC]` oder Fehler ist, und jegliche zwischenzeitige Programmänderung ein Wiederstarten unmöglich macht.

18 `Unknown user function` – unbekannte Anwender-Funktion

Für die gerufene Funktion wurde kein `DEF FN`-Kommando gegeben.

19 `RESUME missing` – `RESUME` fehlt

Das Programmende wurde während einer Fehlerbehandlungsroutine erreicht, z. B. in einer `ON ERROR GOTO` Routine.

20 `Unexpected RESUME` – unerwartetes `RESUME`

`RESUME` ist nur während einer Fehlerbehandlung erlaubt, z. B. in einer `ON ERROR GOTO` Routine

21 `Direct command found` – Direktkommando gefunden

Eine Zeile ohne Zeilennummer wurde gefunden, während ein Programm von Cassette geladen wurde.

22 `Operand missing` – Operand fehlt

BASIC hat eine unvollständige Anweisung gefunden.

23 `Line too long` – Zeile zu lang

Eine Zeile wird im internen BASIC-Format zu groß

24 `EOF met` – `EOF` angetroffen

Es wurde versucht, eine Cassetten-Eingabedatei weiter zu lesen, obwohl deren Dateiende (end of file) schon erreicht ist.

25 File type error – Dateityp fehlerhaft

Die gelesene Cassettendatei hat einen unzulässigen Typ. OPENIN ist nur für ASCII-Textdateien erlaubt. LOAD, RUN, usw. sind nur für Dateien erlaubt, die mit SAVE erstellt wurden.

26 NEXT missing – NEXT fehlt

Ein FOR- Kommando wurde angegeben, ein dazu passendes NEXT kann nicht gefunden werden.

27 File already open – Datei schon geöffnet

Ein OPENIN- oder OPENOUT-Kommando wurde gegeben, obwohl die früher schon eröffnete Datei noch nicht wieder geschlossen wurde.

28 Unknown command – Kommando unbekannt

Ein externes Kommando ist im BASIC unbekannt.

29 WEND missing – WEND fehlt

Ein WHILE Kommando wurde angegeben, ein dazu passendes WEND kann nicht gefunden werden.

30 Unexpected WEND – unerwartetes WEND

Ein WEND wurde außerhalb einer WHILE Schleife gefunden, oder ein WEND paßt nicht zu der aktuellen WHILE Schleife.

BASIC Schlüsselwörter

Die folgenden BASIC Schlüsselwörter sind reserviert und dürfen nicht als Variablennamen verwendet werden.

ABS, AFTER, AND, ASC, ATN, AUTO

BIN$, BORDER
CALL, CAT, CHAIN, CHR$, CINT, CLEAR, CLG, CLOSEIN, CLOSEOUT, CLS, CONT, COS, CREAL

DATA, DEF, DEFINT, DEFREAL, DEFSTR, DEG, DELETE, DI, DIM, DRAW, DRAWR

EDIT, EI, ELSE, END, ENT, ENV, EOF, ERASE, ERL, ERR, ERROR, EVERY, EXP

FIX, FN, FOR, FRE

GOSUB, GOTO

HEX$, HIMEM

IF, INK, INKEY, INKEY$, INP, INPUT, INSTR, INT

JOY

KEY

LEFT$, LEN, LET, LINE, LIST, LOAD, LOCATE, LOG, LOG10, LOWER$

MAX, MEMORY, MERGE, MID$, MIN, MOD, MODE, MOVE, MOVER

NEXT, NEW, NOT

ON, ON BREAK, ON ERROR GOTO, ON SQ, OPENIN, OPENOUT, OR, ORIGIN, OUT

PAPER, PEEK, PEN, PI, PLOT, PLOTR, POKE, POS, PRINT

RAD, RANDOMIZE, READ, RELEASE, REM, REMAIN, RENUM, RESTORE, RESUME, RETURN, RIGHT$, RND, ROUND, RUN

SAVE, SGN, SIN, SOUND, SPACE$, SPC, SPEED, SQ, SQR, STEP, STOP, STR$, STRING$, SWAP, SYMBOL

TAB, TAG, TAGOFF, TAN, TEST, TESTR, THEN, TIME, TO, TROFF, TRON

UNT, UPPER$, USING

VAL, VPOS

WAIT, WEND, WHILE, WIDTH, WINDOW, WRITE

XOR, XPOS

YPOS

ZONE

Index

ABS K8.3
Addition G2.11
AFTER K8.3 K10.1
AND K4.18
Anschlüsse Anh. 5.1
Arrays K4.13
Arithmetik K2.10
ASC K8.4
ASCII K1.7 Anh. 3.1 Anh. 3.14
Assembler K9.5
ATN K8.4
AUTO K4.16 K8.4

BASIC G2.4 K8.1 Anh. 1.2
Anh. 8.5
Bedingungen (u. logische Ausdrücke)
K4.3 K4.18
Befehle, Schlüsselwörter G2.4 K8.1
Anh. 8.5
Begriffserklärungen Anh. 1.B.1
Betriebssystem K9.4
BIN$ K8.4
Blinkende Farben G3.6
BORDER G3.2 K4.12 K8.5
Anh. 4.5

CALL K8.5
CAPS LOCK Taste G2.2
Cassette G1.7 K2.1
CAT K2.7 K8.5
CHAIN, CHAIN MERGE K8.6
CHR$ G3.8 K1.7 K8.6
CINT K8.6
CLEAR K8.7
CLG K8.7
CLOSEIN K8.7
CLOSEOUT K8.7
CLR Taste G2.2
CLS G2.4 K8.8
CONT K4.6 K8.8
Copy Cursor G2.8 K1.15
COPY Taste G2.8 K1.15
COS K8.8
CREAL K8.9
CTRL Taste G2.2 K1.7 K9.2
Cursor G1.2 K1.12 K5.7 K9.1
Anh. 4.3

DATA K4.14 K8.9
Datacorder G1.7 K2.1
DEF FN K8.10
DEFINT, DEFREAL, DEFSTR K8.10
DEG K8.10
DEL Taste G2.1
DELETE K8.11
DI K8.11
DIM K4.13 K8.12
Disketten-Laufwerk Anh. 1.3
Anh. 4.4 Anh. 5.2
Division G2.10
DRAW G3.11 K8.12
DRAWR K8.12
Drucker (Printer) K7.2 Anh. 1.3
Anh. 4.4 Anh. 5.2

EDIT G2.7 K1.16 K8.13
Edit-Korrekturmethode G2.7 K1.14
EI K8.13
ELSE K8.19
END G2.9 K8.13
ENT G3.18 K6.9 K8.13
ENTER Taste G2.1 K1.8
ENV G3.17 K6.8 K8.14
EOF K8.16
ERASE K8.16
ERL, ERR K8.16
ERROR K8.17
Erweiterter Zeichensatz Anh. 4.2
ErweiterungsROMs Anh. 1.3
Anh. 4.4
Erweiterungszeichen Anh. 3.15
ESC Taste G2.3
EVERY K8.17 K10.2
EXP K8.17

Farben G3.1 K5.1 Anh. 4.3
Anh. 4.6
Farbmonitor G1.3 K1.2
Fehlercodes, -nummern und
-meldungen Anh. 8.1
Fernsehgerät G1.5
F.F. Taste K2.2
FIX K8.17
Flush-Tonkanäle K6.7
FOR G2.9 K8.18
FRE K8.18

Gemischte Kalkulationen G2.12 K4.2
Geräusche G3.20
GOSUB G3.14 K8.18
GOTO G2.5 K8.18
Graphik G3.8 K7.3
Grünmonitor G1.1 K1.3

Hardware Anh. 4.4
Helligkeitseinstellung G1.2 G1.4 K1.2
HEX$ K8.19
HIMEN K8.19

IF G2.8 K4.11 K8.19
INK G3.2 K8.20
INKEY K8.20
INP K8.21
INPUT G2.6 K8.21
INSTR K8.22
INT K8.22
I/O (E/A) G3.16 Anh. 4.7 Anh. 5.1

JOY K8.22
Joysticks G1.7 K7.1 Anh. 3.14 Anh. 3.16

KEY DEF K8.23
Kontrasteinstellung G1.2 K1.3
Kontrollzeichen K9.1
Koordinaten G3.11 Anh. 4.4
Kubikwurzel G2.11

Laden der Welcome-Cassette G1.8 K2.4
Laden von Cassetten G1.9 K2.3 K8.25
Lautstärkenstellung G3.16 K4.15
Lautstärkenhüllkurve G3.17 K6.8 K8.14
LEFT$ K8.23
LEN K8.24
Lesefehler K2.8
LET K8.24
LINE INPUT K8.24
LIST G2.5 K1.12 K8.24
LOAD K8.25
LOCATE G3.8 K4.11 K8.25
LOG K8.25
LOG 10 K8.26
Logische Ausdrücke K4.3 K4.18
LOWER$ K8.26

MAX K8.26
Mehrstimmiger Klang Anh. 4.3
MEMORY K8.19 K8.26
MERGE K8.27
MID$ K8.27
MIN K8.27
MOD K4.2
MODE G3.1 K5.3 K8.28 Anh. 4.2
Modulator/Stromversorgung (MP1) G1.5 K1.5
MOVE K8.28
MOVER K8.28
Multiplikation G2.10
Musiknoten Anh. 7.1
Musik Planer Anh. 7.4

Netzschalter G1.2 G1.3 G1.6
NEW G3.13 K8.28
NEXT G2.9 K8.29
NOT K4.20
Notation K8.1

ON BREAK GOSUB K8.29
ON BREAK STOP K8.30
ON ERROR GOTO K8.30 Anh. 8.1
ON GOSUB, ON GOTO K8.29
ON SQ GOSUB K6.10 K8.30
OPENIN K8.31
OPENOUT K8.31
Operatoren K4.3
OR K4.18
ORIGIN F3.14 K8.32
OUT K8.32

PAPER G3.2 K8.33
PAUSE Taste K2.2
PEEK K8.33
PEN G3.2 K8.34
PI K8.34
PLOT G3.11 K8.35
PLOTR K8.35
POKE K8.36
POS K8.36
Potenzen G2.12 G2.13
PRINT G2.4 G3.4 K8.36 K8.54
PRINT SPC K4.16
PRINT TAB K3.6
PRINT USING K3.6

RAD K8.37
RAM Anh. 4.6
RANDOMIZE K8.37
READ K4.14 K8.9 K8.37
REC Taste K2.2
RELEASE K6.7 K6.11 K8.38
REM K5.4 K8.38
REMAIN K8.38 K10.3
Rendezvous für die Tonkanäle K6.4
RENUM K4.8 K5.4 K8.39
RESTORE K8.39
RESUME K8.39
RETURN G3.14 K8.40

REW Taste K2.2
RIGHT$ K8.40
RND K8.40
ROUND K8.41
RUN G2.3 K8.41
Rückseitige Anschlüsse Anh. 5.1

SAVE G1.11 K2.6 K8.42
SGN K8.42
SHIFT Taste G2.1
Schreibschutz K2.3
SIN K8.42
Sound G3.16 K6.1 K8.43
Anh. 1.3 Anh. 7.1
SPACE$ K8.43
SPC K4.16 K8.36
SPEED INK K4.13 K8.43
SPEED KEY K8.44
Speedload K2.5
SPEEDWRITE K2.6 K8.44
Speicherbelegung Anh. 4.6
SQ K6.10 K8.45
SQR K8.45
Square Root G2.11
STEP G2.10 K8.18
Stereo G3.16 K6.4 Anh. 1.3
STOP K8.45
STOP/EJECT Taste K2.2
STR$ K8.46
STRING$ K8.46
String-(Zeichenfolge)Variable G2.6
K4.7
Subtraktion G2.11
Supersafe-Laden K2.5
SYMBOL K8.46
SYMBOL AFTER K8.47
Syntax Error G2.3 K4.1 Anh. 1.5
Anh. 2.2 Anh. 8.1

TAB K3.6
TAB Taste K3.6
TAG K8.47
TAGOFF K8.47
TAN K8.48
Tastatur G2.1 K1.8 Anh. 3.14
Anh. 4.7
TEST K8.48
TESTR K8.48
Text/Window Planer Anh. 6.1
THEN G2.9 K4.11 K8.19
TIME K8.48 K8.51
TO G2.10 K8.18
Ton-Hüllkurve G3.18 K6.9 K8.14
Tonkanal-Haltestellen K6.5
Tonvariationsplaner Anh. 7.4
Transparentdarstellung K5.2
Trennzeichen K1.7 K3.2 K4.1
TRON, TROFF K8.49
Typenkennzeichen K4.6

UNT K8.49
UPPER$ K8.49
Anwenderdefinierte Taste Anh. 4.2
Unterbrechungen K9.5 K10.1
USER PORT G1.7 K7.1 Anh. 5.1
USING K3.6

VAL K8.49
Variablen G2.6 K4.1 K4.6
Vertikale Verschiebung G1.2 K1.3
VPOS K8.50

WAIT K8.50
Welcome Cassette G1.7 K2.4
WEND K8.51
WHILE K8.51
WIDTH K8.52
WINDOW K5.10 K8.52 Anh. 4.3
Window Planer Anh. 6.1
WINDOW SWAP K5.10 K8.52
WRITE K8.52

XOR K4.18
XPOS K8.53

YPOS K8.53

Zeichen (Characters) K1.9 Anh. 3.1
ZONE K3.6 K8.53
Zurücksetzen G1.9 K1.2
Zusätzliche ROMs Anh. 4.6